



Volumes bereitstellen und verwalten

Trident

NetApp
October 29, 2026

Inhalt

Volumes bereitstellen und verwalten	1
Bereitstellung eines Trident Volume	1
Überblick	1
PVC erstellen	1
Ein Volume verschieben	3
Anforderungen	4
So funktioniert es	4
Ein Volume verschieben	4
Trident Volumes erweitern	7
Ein iSCSI-Volume erweitern	7
Ein FC-Volume erweitern	11
Ein NFS-Volume erweitern	14
Grenzen des RWX NVMe-Subsystems verstehen	17
Verstehen Sie das 64-Knoten-Limit	17
NVMe-Subsystemmodelle verstehen	17
Fehlersymptome identifizieren	18
Beheben Sie Subsystem-Limit-Fehler	18
Aktualisieren Sie Trident, um das Super-Subsystem-Modell anzuwenden	18
Controller-Skalierbarkeit	18
Schlüsselbegriffe und Definitionen	19
Standardverhalten (serieller Modus)	19
Unterstützung der Controller-Skalierbarkeit	19
Wie enableConcurrency funktioniert	20
Bevor Sie aktivieren	20
Controller-Skalierbarkeit aktivieren	20
Parallelitätsverhalten	23
Vorbehalte und Einschränkungen	23
Empfehlungen	23
Automatische Volumenerweiterung	24
Anforderungen	24
Einschränkungen	24
Volumes mit Autogrow-Richtlinie bereitstellen	25
Erstellen Sie eine Autogrow-Richtlinie	25
Erstellen Sie eine Autogrow-Richtlinie	26
Richtlinienstatus	26
Ordnen Sie eine Richtlinie einer StorageClass zu	27
Richtlinienvorrang	28
Konfigurationsbeispiele	28
Autogrow-Richtlinien verwalten	30
Autogrow-Richtlinien anzeigen	31
Eine Autogrow-Richtlinie aktualisieren	31
Löschen einer Autogrow-Richtlinie	32
Autogrow-Richtliniennutzung überwachen	33

Unterstützte Protokolle	33
Unterstützte Backends	33
Bekannte Einschränkungen	34
Häufig gestellte Fragen	34
Importieren von Trident Volumes	39
Überblick und Überlegungen	40
Importieren Sie ein Volume	41
Beispiele	42
ONTAP SAN-economy Beispiele	48
Anpassung von Trident Volume-Namen und Labels	51
Bevor Sie beginnen	51
Einschränkungen	52
Wichtige Verhaltensweisen von anpassbaren Volumennamen	52
Backend-Konfigurationsbeispiele mit Namensvorlage und Labels	52
Beispiele für Namensvorlagen	53
Zu berücksichtigende Punkte	53
Ein Trident NFS Volume über Namespaces hinweg freigeben	54
Features	54
Schnellstart	55
Konfigurieren Sie die Quell- und Ziel-Namespaces	56
Löschen eines freigegebenen Volumes	57
Verwenden Sie <code>tridentctl get</code> zur Abfrage untergeordneter Volumes	57
Einschränkungen	58
Für weitere Informationen	58
Klonen von Trident Volumes über Namespaces hinweg	58
Voraussetzungen	58
Schnellstart	58
Konfigurieren Sie die Quell- und Ziel-Namespaces	59
Einschränkungen	60
Volumes replizieren mit SnapMirror	60
Replikationsvoraussetzungen	61
Erstellen Sie ein gespiegeltes PVC	62
Volumenreplikationszustände	65
Sekundäres PVC während eines ungeplanten Failovers promoten	65
Sekundäre PVC während eines geplanten Failovers promoten	65
Eine Spiegelbeziehung nach einem Failover wiederherstellen	66
Zusätzliche Operationen	66
Spiegelbeziehungen aktualisieren, wenn ONTAP online ist	67
Spiegelbeziehungen aktualisieren, wenn ONTAP offline ist	67
CSI-Topologie verwenden	67
Überblick	67
Schritt 1: Erstellen eines topologiebewussten Backends	69
Schritt 2: Definieren Sie StorageClasses, die topologiebewusst sind	71
Schritt 3: Eine PVC erstellen und verwenden	71
Backends aktualisieren, um <code>supportedTopologies</code> einzuschließen	74

Weitere Informationen	74
Arbeiten mit Trident Snapshots	74
Überblick	74
Erstellen Sie einen Volume-Snapshot	75
Erstellen Sie eine PVC aus einem Volume-Snapshot	76
Importieren eines Volume-Snapshots	77
Stellen Sie Volumendaten mithilfe von Snapshots wieder her	79
In-Place-Volume-Wiederherstellung aus einem Snapshot	79
Löschen eines PV mit zugehörigen Snapshots	81
Einen Volume Snapshot Controller bereitstellen	81
Verwandte Links	82
Arbeiten mit Trident Volume-Gruppen-Snapshots	82
Snapshots von Volumengruppen erstellen	82
Volumendaten mithilfe eines Gruppen-Snapshots wiederherstellen	84
In-Place-Volume-Wiederherstellung aus einem Snapshot	85
Löschen eines PV mit zugehörigen Gruppensnapshots	85
Einen Volume Snapshot Controller bereitstellen	85
Verwandte Links	86

Volumes bereitstellen und verwalten

Bereitstellung eines Trident Volume

Erstellen Sie einen PersistentVolumeClaim (PVC), der die konfigurierte Kubernetes-StorageClass verwendet, um Zugriff auf das PV anzufordern. Anschließend können Sie das PV in einen Pod einbinden.

Überblick

Ein "*PersistentVolumeClaim*" (PVC) ist eine Anfrage für den Zugriff auf das PersistentVolume im Cluster.

Die PVC kann so konfiguriert werden, dass sie Speicherplatz einer bestimmten Größe oder eines bestimmten Zugriffsmodus anfordert. Mithilfe der zugehörigen StorageClass kann der Clusteradministrator mehr als nur PersistentVolume-Größe und Zugriffsmodus steuern – zum Beispiel Leistung oder Servicelevel.

Nachdem Sie das PVC erstellt haben, können Sie das Volume in einem Pod einbinden.

PVC erstellen

Schritte

1. Die PVC wird erstellt.

```
kubectl create -f pvc.yaml
```

2. Überprüfen Sie den PVC-Status.

```
kubectl get pvc
```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
pvc-storage	Bound	pv-name	1Gi	RWO		5m

1. Mountieren Sie das Volume in einem Pod.

```
kubectl create -f pv-pod.yaml
```



Sie können den Fortschritt mit `kubectl get pod --watch` überwachen.

2. Überprüfen Sie, ob das Volume auf `/my/mount/path` eingebunden ist.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. Sie können den Pod nun löschen. Die Pod-Anwendung existiert dann nicht mehr, aber das Volume bleibt erhalten.

```
kubectl delete pod pv-pod
```

Beispielmanifeste

▼ PersistentVolumeClaim-Beispielmanifeste

Diese Beispiele zeigen grundlegende PVC-Konfigurationsoptionen.

PVC mit RWO-Zugriff

Dieses Beispiel zeigt eine einfache PVC mit RWO-Zugriff, die mit einer StorageClass namens `basic-csi` verknüpft ist.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC mit NVMe/TCP

Dieses Beispiel zeigt eine einfache PVC für NVMe/TCP mit RWO-Zugriff, die einer StorageClass namens `protection-gold` zugeordnet ist.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san-nvme
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 300Mi
  storageClassName: protection-gold
```

▼ Pod-Manifestbeispiele

Diese Beispiele zeigen grundlegende Konfigurationen, um das PVC an ein Pod anzuhängen.

Grundkonfiguration

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
        claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

Grundlegende NVMe/TCP-Konfiguration

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Siehe ["Kubernetes- und Trident-Objekte"](#) für Details dazu, wie Storage-Klassen mit dem PersistentVolumeClaim interagieren und Parameter zur Steuerung, wie Trident Volumes bereitstellt.

Ein Volume verschieben

Ab Trident 26.06 kann ein Speicheradministrator ein iSCSI-Volume unterbrechungsfrei

auf einen anderen ONTAP-Knoten und ein anderes Aggregat innerhalb desselben ONTAP Clusters verschieben, während die darauf basierende Arbeitslast weiterläuft. Die Verschiebung wird deklarativ mit einer `TridentVolumeMove` benutzerdefinierten Ressource oder dem `tridentctl` Äquivalent angefordert, und Trident koordiniert ONTAP und den Host, sodass die I/O niemals unterbrochen wird.

Anforderungen

Volume Move gilt nur für den `ontap-san` Treiber für iSCSI-Volumes. NVMe und NAS Protokolle werden nicht unterstützt.

Bevor Sie ein Volume verschieben, muss sichergestellt sein, dass die folgenden Voraussetzungen erfüllt sind:

- Das Backend verwendet den `ontap-san` Treiber mit der ONTAP REST API.
- Der ONTAP Cluster verfügt über mindestens zwei Knoten.
- Sie verfügen über einen der folgenden Berechtigungssätze:
 - ONTAP 9.19.1 oder höher, wenn die dedizierte benutzerdefinierte ONTAP Rolle verwendet wird.
 - Cluster-Administrator-Anmeldeinformationen in früheren ONTAP Releases.

So funktioniert es

Trident verschiebt das Volume (LUN) zwischen zwei Standorten innerhalb desselben ONTAP Clusters. Bevor die Daten verschoben werden, fügt Trident dem Ziel neben den bestehenden Pfaden neue iSCSI-Pfade hinzu, sodass das Multipath-Gerät des Hosts jederzeit erreichbar bleibt. Nachdem ONTAP die Verschiebung abgeschlossen hat, entfernt Trident die veralteten Quellpfade.

Trident steuert den Vorgang durch eine geordnete Reihe von Zuständen, die in der `TridentVolumeMove` Ressource sichtbar sind, sodass der Fortschritt des Vorgangs verfolgt werden kann.

Ein Volume verschieben

Das Volumen und die Standorte identifizieren

1. Den Namen des zu verschiebenden PV ermitteln. Der PV-Name ist auch der Name der Move-Ressource.
2. Im ONTAP Cluster den aktuellen (Quell-)Knoten und das Aggregat für dieses Volume notieren und den Zielknoten sowie das Ziel-Aggregat auswählen.

Starten Sie den Umzug

Beide Methoden können verwendet werden. Beide erstellen dieselbe `TridentVolumeMove` Ressource, deren Name der PV-Name ist.

Verwendung von `tridentctl`

```
tridentctl create volume-move \  
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \  
  --source-node ontap-node-01 \  
  --source-pool aggr1_node01 \  
  --target-node ontap-node-02 \  
  --target-pool aggr2_node02
```

```
--target-node ontap-node-02 \  
--target-pool aggr1_node02 \  
-n trident
```

tvm ist ein Alias für volume-move. --delete-after-success 10m hinzufügen, damit die Ressource nach erfolgreichem Verschieben automatisch entfernt wird.

Verwendung einer TridentVolumeMove Ressource

```
apiVersion: trident.netapp.io/v1  
kind: TridentVolumeMove  
metadata:  
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671    # the PV to move  
  namespace: trident  
spec:  
  sourceNode: ontap-node-01  
  sourcePool: aggr1_node01  
  targetNode: ontap-node-02  
  targetPool: aggr1_node02  
  # deleteAfterSuccess: 10m    # optional
```

Ressource anwenden:

```
kubectl apply -f tridentvolumemove.yaml
```

Spezifikationsfelder

sourceNode / targetNode

ONTAP Knotennamen (nicht UUIDs).

sourcePool / targetPool

ONTAP Aggregatnamen.

deleteAfterSuccess

Optionale Dauer, wie 30s oder 10m. Auf 0 setzen, um die Ressource sofort zu löschen, oder weglassen, um die Ressource beizubehalten.

Die Verschiebung überwachen

Der Fortschritt kann anhand des tvms`Kurznamens oder `tridentvolumemove verfolgt werden:

```
kubectl get tridentvolumemove -n trident
```

Der aktuelle Status und der Fortschritt pro Knoten werden angezeigt, wenn die Ressource beschrieben wird:

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

Das Volume kann während des Moving-Zustands mehrere gleichzeitige Anhänge haben, da ONTAP für die Verschiebung eines großen Volumes einige Zeit benötigen kann. Für eine reibungslose Verschiebung koordiniert Trident die Abläufe zwischen dem Trident Controller und dem Trident Node. Jeder Zustand führt die folgenden Schritte in der angegebenen Reihenfolge aus und endet entweder in Succeeded oder Failed.

Status	Was geschieht
Pending	Trident akzeptiert und validiert die Umzugsanfrage und sammelt die benötigten Informationen, bevor Änderungen am Speicher vorgenommen werden.
ControllerStaging	Trident bereitet den Zielort vor. Es ordnet die Speicherschnittstellen des Zielknotens der LUN zu und aktualisiert Kubernetes, sodass ein zukünftiger Neustart des Knotens die Verbindung zum richtigen Standort wiederherstellt.
NodeStaging	Auf jedem Host, auf dem das Volume eingebunden ist, fügt Trident die neuen Speicherpfade vom Zielknoten neben den bestehenden Pfaden hinzu, sodass die E/A des Pods über beide Pfadsätze hinweg fortgesetzt wird.
Moving	ONTAP kopiert und überträgt die Volumendaten auf das Zielaggregat. Dies ist die längste Phase, und ONTAP übernimmt sie vollständig, während Trident den Vorgang abfragt, bis er abgeschlossen ist. Neue Veröffentlichungen sind während dieses Status zulässig.
ControllerUnstaging	Die Datenübertragung ist abgeschlossen. Trident aktualisiert Kubernetes, um die Quellpfade aus dem gewünschten Zustand zu entfernen, und hebt anschließend die Zuordnung der Speicherschnittstellen des Quellknotens von der LUN auf.
NodeUnstaging	Auf jedem Host entfernt Trident die veralteten Quellpfade. Dem Host verbleiben nur noch die Zielpfade, und das Multipath-Gerät verarbeitet weiterhin E/A ohne Unterbrechung.
Succeeded	Das Volume ist vollständig verschoben und am Ziel online. Trident behält die TridentVolumeMove Ressource bei oder löscht sie automatisch, wenn Sie deleteAfterSuccess festlegen.
Failed	Ein nicht behebbarer Fehler hat den Verschiebevorgang gestoppt oder dessen Start verhindert. Trident setzt den Volume-Status zurück, sodass der normale Betrieb fortgesetzt werden kann, und behält die Ressource mit einer Fehlermeldung. Die Meldung sollte überprüft, die Ursache behoben, die Ressource gelöscht und der Vorgang erneut versucht werden.

Die `.status.attachments``Liste zeigt den Fortschritt jedes angeschlossenen Knotens in seinem ``state``Feld:

Anhangsstatus	Bedeutung
Pending	Knoten als verbunden identifiziert; die Arbeit hat noch nicht begonnen.
Bridged	Neue Zielpfade werden neben den bestehenden Pfaden hinzugefügt.

Anhangsstatus	Bedeutung
Migrated	Der Datenverkehr wird jetzt über die neuen Pfade bereitgestellt.
Cleaned	Veraltete Quellpfade vom Knoten entfernt.
Detached	Der Knoten wurde während der Bewegung abgetrennt und von der weiteren Betrachtung ausgeschlossen, während Trident prüft, ob die Verbindung bereit ist, mit dem nächsten Schritt fortzufahren.
Failed	Terminalfehler auf diesem Knoten. Die Meldung anzeigen.

Die Umzugsressource bereinigen

Wenn `deleteAfterSuccess` festgelegt ist, entfernt Trident die Ressource automatisch. Andernfalls kann sie manuell gelöscht werden. Bevor eine weitere Verschiebung auf demselben Volume angefordert wird, ist sicherzustellen, dass für dieses Volume keine `TridentVolumeMove` Ressource im `Succeeded`- oder `Failed`-Status vorhanden ist.



Trident löscht fehlgeschlagene Verschiebungen niemals automatisch. Trident speichert eine `Failed` Ressource als Prüfprotokoll. Nachdem die Ursache behoben wurde, sollte die Ressource manuell gelöscht werden, bevor ein erneuter Versuch erfolgt.

Trident Volumes erweitern

Trident ermöglicht Kubernetes-Nutzern die nachträgliche Erweiterung ihrer Volumes. Finden Sie Informationen zu den erforderlichen Konfigurationen, um iSCSI-, NFS-, SMB-, NVMe/TCP- und FC-Volumes zu erweitern.

Ein iSCSI-Volume erweitern

Sie können ein iSCSI Persistent Volume (PV) mithilfe des CSI provisioner erweitern.



Die iSCSI-Volume-Erweiterung wird von den `ontap-san`, `ontap-san-economy`, `solidfire-san` Treibern unterstützt und erfordert Kubernetes 1.16 oder höher.

Schritt 1: Konfigurieren Sie die StorageClass, um die Volumenerweiterung zu unterstützen

Bearbeiten Sie die StorageClass-Definition, um das `allowVolumeExpansion` Feld auf `true` zu setzen.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
```

```
backendType: "ontap-san"
allowVolumeExpansion: True
```

Bearbeiten Sie eine bereits bestehende StorageClass, um den `allowVolumeExpansion` Parameter hinzuzufügen.

Schritt 2: Erstellen Sie eine PVC mit der StorageClass, die Sie erstellt haben

Bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage`, um die neu gewünschte Größe anzugeben, die größer als die ursprüngliche Größe sein muss.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident erstellt ein Persistent Volume (PV) und ordnet es diesem Persistent Volume Claim (PVC) zu.

```
kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO           ontap-san    8s

kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS  CLAIM                                STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi      RWO
Delete          Bound    default/san-pvc                    ontap-san    10s
```

Schritt 3: Definieren Sie ein Pod, das das PVC anbindet

Befestigen Sie das PV an einem Pod, damit es in der Größe angepasst werden kann. Es gibt zwei Szenarien, wenn die Größe eines iSCSI PV geändert wird:

- Wenn das PV an einen Pod angehängt ist, erweitert Trident das Volume auf dem Storage-Backend, scannt das Gerät erneut und passt die Größe des Dateisystems an.
- Beim Versuch, ein nicht zugeordnetes PV zu vergrößern, erweitert Trident das Volume im Storage-Backend. Nachdem das PVC an einen Pod gebunden wurde, scannt Trident das Gerät erneut und passt die Größe des Dateisystems an. Kubernetes aktualisiert dann die PVC-Größe, nachdem der Erweiterungsvorgang erfolgreich abgeschlossen wurde.

In diesem Beispiel wird ein Pod erstellt, der die `san-pvc` verwendet.

```
kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
ubuntu-pod    1/1     Running   0           65s

kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
               csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Schritt 4: PV erweitern

Um die Größe des erstellten PV von 1Gi auf 2Gi zu ändern, bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage` auf 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
# ignored,
# and an empty file will abort the edit. If an error occurs while saving
# this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
```

```

metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
# ...

```

Schritt 5: Validieren Sie die Erweiterung

Sie können überprüfen, ob die Erweiterung korrekt funktioniert hat, indem Sie die Größe des PVC, PV und des Trident volume kontrollieren:

```

kubectl get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound       pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO           ontap-san    11m
kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM                                STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi       RWO
Delete       Bound     default/san-pvc  ontap-san                                12m
tridentctl get volumes -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE  | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID  | STATE | MANAGED  |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san  |
| block      | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true      |
+-----+-----+-----+-----+

```

Ein FC-Volume erweitern

Sie können ein FC Persistent Volume (PV) mithilfe des CSI provisioner erweitern.



FC-Volume-Erweiterung wird vom `ontap-san` Treiber unterstützt und erfordert Kubernetes 1.16 und höher.

Schritt 1: Konfigurieren Sie die StorageClass, um die Volumenerweiterung zu unterstützen

Bearbeiten Sie die StorageClass-Definition, um das `allowVolumeExpansion` Feld auf `true` zu setzen.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Bearbeiten Sie eine bereits bestehende StorageClass, um den `allowVolumeExpansion` Parameter hinzuzufügen.

Schritt 2: Erstellen Sie eine PVC mit der StorageClass, die Sie erstellt haben

Bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage`, um die neu gewünschte Größe anzugeben, die größer als die ursprüngliche Größe sein muss.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
```

```
storage: 1Gi
storageClassName: ontap-san
```

Trident erstellt ein Persistent Volume (PV) und ordnet es diesem Persistent Volume Claim (PVC) zu.

```
kubectl get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM                                STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi      RWO
Delete          Bound    default/san-pvc                    ontap-san      10s
```

Schritt 3: Definieren Sie ein Pod, das das PVC anbindet

Befestigen Sie das PV an einem Pod, damit es in der Größe angepasst werden kann. Es gibt zwei Szenarien, wenn ein FC-PV in der Größe geändert wird:

- Wenn das PV an einen Pod angehängt ist, erweitert Trident das Volume auf dem Storage-Backend, scannt das Gerät erneut und passt die Größe des Dateisystems an.
- Beim Versuch, ein nicht zugeordnetes PV zu vergrößern, erweitert Trident das Volume im Storage-Backend. Nachdem das PVC an einen Pod gebunden wurde, scannt Trident das Gerät erneut und passt die Größe des Dateisystems an. Kubernetes aktualisiert dann die PVC-Größe, nachdem der Erweiterungsvorgang erfolgreich abgeschlossen wurde.

In diesem Beispiel wird ein Pod erstellt, der die `san-pvc` verwendet.

```
kubectl get pod
NAME          READY    STATUS    RESTARTS   AGE
ubuntu-pod    1/1     Running   0          65s

kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
```

```
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Schritt 4: PV erweitern

Um die Größe des erstellten PV von 1Gi auf 2Gi zu ändern, bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage` auf 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
# ...
```

Schritt 5: Validieren Sie die Erweiterung

Sie können überprüfen, ob die Erweiterung korrekt funktioniert hat, indem Sie die Größe des PVC, PV und des Trident volume kontrollieren:

```
kubectl get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO           ontap-san    11m
kubectl get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY STATUS    CLAIM          STORAGECLASS  REASON    AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi        RWO
Delete              Bound      default/san-pvc  ontap-san    12m
tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san    |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Ein NFS-Volume erweitern

Trident unterstützt die Volume-Erweiterung für NFS PVs, die auf `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` und `azure-netapp-files` Backends bereitgestellt werden.

Schritt 1: Konfigurieren Sie die StorageClass, um die Volumenerweiterung zu unterstützen

Um die Größe eines NFS-PV zu ändern, muss der Administrator zunächst die Speicherklasse so konfigurieren, dass eine Volumenerweiterung möglich ist, indem er das `allowVolumeExpansion` Feld auf `true` setzt:

```
cat storageclass-ontapnas.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
```

```
backendType: ontap-nas
allowVolumeExpansion: true
```

Falls Sie bereits eine Speicherklasse ohne diese Option erstellt haben, können Sie die bestehende Speicherklasse einfach bearbeiten, indem Sie `kubectl edit storageclass` verwenden, um die Volumenerweiterung zu ermöglichen.

Schritt 2: Erstellen Sie eine PVC mit der StorageClass, die Sie erstellt haben

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident sollte eine 20 MiB NFS PV für diese PVC erstellen:

```
kubectl get pvc
NAME                STATUS    VOLUME
CAPACITY            ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb        Bound       pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi
RWO                  ontapnas      9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY  ACCESS MODES
RECLAIM POLICY      STATUS    CLAIM                STORAGECLASS  REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi      RWO
Delete              Bound     default/ontapnas20mb  ontapnas
2m42s
```

Schritt 3: PV erweitern

Um die neu erstellte 20 MiB PV auf 1 GiB zu skalieren, bearbeiten Sie die PVC und setzen Sie `spec.resources.requests.storage` auf 1 GiB:

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

Schritt 4: Validieren Sie die Erweiterung

Sie können validieren, ob die Größenänderung korrekt funktioniert hat, indem Sie die Größe des PVC, PV und des Trident-Volumes überprüfen:

```
kubectl get pvc ontapnas20mb
NAME          STATUS    VOLUME
CAPACITY     ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb  Bound      pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  1Gi
RWO           ontapnas      4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME          CAPACITY  ACCESS MODES
```

```

RECLAIM POLICY    STATUS    CLAIM                                STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7    1Gi            RWO
Delete                Bound    default/ontapnas20mb    ontapnas
5m35s

```

```

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID  | STATE | MANAGED |
+-----+-----+-----+-----+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas |
file          | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

Grenzen des RWX NVMe-Subsystems verstehen

ReadWriteMany (RWX)-Volumes, die das NVMe-Protokoll verwenden, haben eine Skalierbarkeitsgrenze von 64 Knoten pro Volume. Im Folgenden werden die Einschränkungen erläutert, die Architektur des beteiligten NVMe-Subsystems erklärt und die erforderlichen Schritte zur Behebung aufgezeigt.

Verstehen Sie das 64-Knoten-Limit

Wenn Sie planen, ReadWriteMany (RWX)-Volumes mit dem NVMe-Protokoll zu verwenden, kann ein einzelnes RWX-NVMe-Volume von nicht mehr als 64 Knoten in einem Kubernetes-Cluster eingebunden werden.

Planen Sie keine Workloads ein, die dieselbe RWX NVMe PersistentVolumeClaim auf mehr als 64 Knoten einbinden.

Diese Einschränkung gilt nur für RWX-Volumes, die das NVMe-Protokoll verwenden.

NVMe-Subsystemmodelle verstehen

Subsystemmodell pro Volumen (Trident Releases früher als 26.02)

In Trident-Versionen vor 26.02 werden RWX-NVMe-Volumes mithilfe eines Subsystemmodells pro Volume bereitgestellt. Jedes RWX-NVMe-Volume ist seinem eigenen dedizierten NVMe-Subsystem auf ONTAP zugeordnet.

Dieses Modell ist einfach, hat aber eine geringere Skalierbarkeitsgrenze. In großen Kubernetes-Clustern werden die Grenzen der Subsystem-Controller schnell erreicht, da jedes RWX-Volume ein eigenes Subsystem belegt.

Super-Subsystem-Modell (eingeführt in Trident 26.02)

Ab Trident 26.02 verwenden RWX NVMe-Volumes ein gemeinsames Super-Subsystem-Modell. Mehrere RWX NVMe-Volumes teilen sich dasselbe NVMe-Subsystem.

Jedes Super-Subsystem unterstützt bis zu 1024 Namensräume (Volumes). Dieses Modell verbessert die Skalierbarkeit für RWX-Workloads erheblich und verringert die Wahrscheinlichkeit, die ONTAP-Subsystem-Grenzen zu erreichen.

Jedes RWX NVMe-Volume unterstützt bis zu 64 Knoten.

Fehlersymptome identifizieren

Wenn Sie RWX NVMe-Volumes in großem Umfang erstellen oder einbinden, können Fehler wie die folgenden auftreten:

```
Maximum number of controllers reached. No more controllers can be created.
```

Dieser Fehler weist darauf hin, dass das ONTAP NVMe-Subsystem-Controller-Limit erreicht wurde.

Beheben Sie Subsystem-Limit-Fehler

Um die Beschränkungen des Subsystems pro Volume zu überwinden und das Super-Subsystem-Modell nutzen zu können, aktualisieren Sie auf Trident 26.02 oder höher.

Aktualisieren Sie Trident, um das Super-Subsystem-Modell anzuwenden

Um das Super-Subsystem-Modell für RWX NVMe-Volumes anzuwenden:

1. Aktualisieren Sie Trident auf Version 26.02 oder höher.
2. Skalieren Sie alle Pods, die RWX NVMe-Volumes verwenden, auf null Replikate herunter.
3. Stellen Sie sicher, dass keine Workloads aktiv RWX NVMe-Volumes verwenden.
4. Skalieren Sie die Pods wieder hoch.

Diese Neustartsequenz stellt sicher, dass RWX NVMe volumes mithilfe des Super-Subsystem-Modells eingebunden werden.

- Diese Einschränkung gilt nur für RWX-Volumes, die das NVMe-Protokoll verwenden.
- Die Beschränkung auf 64 Knoten gilt pro RWX NVMe volume.
- Andere Zugriffsmodi und andere Protokolle sind nicht betroffen.

Controller-Skalierbarkeit

Trident verbessert die Controller Skalierbarkeit durch erhöhte Parallelität über mehrere Speichertreiber hinweg. Bei aktivierter Controller Skalierbarkeit verarbeitet der Trident Controller Speicheroperationen parallel statt sie zu serialisieren, was den Durchsatz in Kubernetes-Umgebungen mit vielen gleichzeitigen Operationen erhöht.

Vor der Bereitstellung sollte ermittelt werden, welche Trident-Treiber die Controller Skalierbarkeit bei

allgemeiner Verfügbarkeit unterstützen und welche Treiber in Trident 26.06 als technische Vorschau verfügbar sind. Dies unterstützt fundierte Entscheidungen zur Bereitstellung und ein effektives Risikomanagement. Die Controller Skalierbarkeit ist standardmäßig deaktiviert.

Schlüsselbegriffe und Definitionen

Controller-Skalierbarkeit

Controller-Skalierbarkeit bezeichnet die Fähigkeit des Trident Controllers, mehrere Speicheroperationen parallel zu verarbeiten, anstatt sie hinter einer einzigen Sperre zu serialisieren. Zu diesen Operationen gehören das Erstellen, Löschen und Ändern der Größe von Volumes, das Erstellen und Löschen von Snapshots, das Veröffentlichen und Aufheben der Veröffentlichung von Volumes sowie die Backend-Verwaltung.

Wenn die Controller-Skalierbarkeit aktiviert ist, werden Vorgänge auf verschiedenen Volumes und Backends gleichzeitig ausgeführt. Dadurch wird der Durchsatz erhöht und die End-to-End-Betriebszeit in Umgebungen mit einer hohen Anzahl gleichzeitiger PersistentVolumeClaim und VolumeSnapshot Vorgänge reduziert.

Standardverhalten (serieller Modus)

Standardmäßig verarbeitet der Trident-Controller Operationen nacheinander. Jede Anfrage zum Erstellen, Löschen, Ändern der Größe oder Erstellen eines Snapshots wird abgeschlossen, bevor Trident die nächste startet. Der serielle Modus ist die unterstützte Standardeinstellung für alle Installationen und erfordert keine Konfiguration.

Der serielle Modus ist für die meisten Arbeitslasten ausreichend. Die Controller Skalierbarkeit sollte nur aktiviert werden, wenn das Verarbeitungsvolumen bei der seriellen Verarbeitung zu einem Rückstau führt.

Unterstützung der Controller-Skalierbarkeit

Trident unterstützt die Skalierbarkeit des Controllers auf unterschiedlichen Reifegraden für verschiedene Speichertreiber.

Allgemeine Verfügbarkeit

Die folgenden Treiber unterstützen die Controller-Skalierbarkeit bei allgemeiner Verfügbarkeit in Trident 26.06:

- `ontap-san`
- `ontap-nas`
- `ontap-nas-economy`
- `ontap-san-economy`
- `google-cloud-netapp-volumes`
- `azure-netapp-files`
- `solidfire-san`



Die `google-cloud-netapp-volumes` und `google-cloud-netapp-volumes-san` Treiber unterscheiden sich. Trident unterstützt nur `google-cloud-netapp-volumes`. `google-cloud-netapp-volumes-san` sollte in Backend-Konfigurationen oder Beispielen nicht verwendet werden.

Technische Vorschau

Der folgende Treiber unterstützt die Controller Skalierbarkeit als technische Vorschaufunktion in Trident 26.06:

- `asa-r2` (SAN und NVMe)

Dieser Treiber hat folgende Einschränkungen:

- Die Controller-Parallelität ist nur zu Evaluierungs- und Testzwecken verfügbar.
- Das Verhalten kann sich von einer Version zur nächsten ändern.
- NetApp empfiehlt die Verwendung in Produktionsumgebungen nicht.

Wie `enableConcurrency` funktioniert

Wenn `enableConcurrency` auf `true` gesetzt wird, wendet Trident parallele Verarbeitung auf alle vom Controller verwalteten Backends an. Die Einstellung gilt für alle Backends gleichzeitig. Eine Aktivierung für einzelne Backends oder einzelne Treiber ist nicht möglich.

Jedes konfigurierte Backend muss einen Treiber aus der Liste der allgemein verfügbaren oder der technischen Vorschauversionen verwenden. Falls ein Backend einen nicht unterstützten Treiber verwendet, startet Trident nicht und fügt auch kein Backend hinzu, das einen nicht unterstützten Treiber verwendet.

Um den Startvorgang wiederherzustellen, müssen alle Backends entfernt oder neu konfiguriert werden, die einen nicht unterstützten Treiber verwenden, oder die Controller Skalierbarkeit wird deaktiviert.

Bevor Sie aktivieren



Bevor Sie die Controller Skalierbarkeit aktivieren, vergewissern Sie sich, dass jedes konfigurierte Backend einen Treiber aus der Liste der allgemein verfügbaren oder technischen Vorschauversionen verwendet. Falls ein Backend einen nicht unterstützten Treiber verwendet, startet Trident nach der Aktivierung der Funktion nicht.

Die folgende Tabelle unterstützt die Entscheidung, ob die Controller Skalierbarkeit aktiviert werden soll.

Wenn Ihre Umgebung	Dann
Verwendet nur unterstützte Treiber und weist einen Rückstau bei Controller-Operationen auf	Controller-Skalierbarkeit aktivieren.
Verwendet einen nicht unterstützten Treiber	Der standardmäßige serielle Modus bleibt erhalten. Die Controller-Skalierbarkeit nicht aktivieren.
Bewältigt geringes Arbeitsaufkommen ohne Rückstände	Der standardmäßige serielle Modus bleibt erhalten.

Controller-Skalierbarkeit aktivieren

Die `enableConcurrency` Konfigurationsoption steuert die Skalierbarkeit des Controllers. Diese Option muss während der Trident Installation oder bei der Aktualisierung einer bestehenden Bereitstellung explizit aktiviert werden.

Trident Operator-Bereitstellung

Um die Controller Skalierbarkeit mit dem Trident Operator zu ermöglichen, `enableConcurrency` auf `true` in der `TridentOrchestrator` benutzerdefinierten Ressource (CR) setzen.

Neuinstallation

Erstellen oder bearbeiten Sie die `TridentOrchestrator` CR und setzen Sie `enableConcurrency` auf `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

Wenden Sie die CR an:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Vorhandene Installation

Patchen Sie die bestehende `TridentOrchestrator` CR, um die Skalierbarkeit des Controllers zu ermöglichen:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Es sollte überprüft werden, ob Trident die Einstellung angewendet hat:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Helm-Bereitstellung

Um die Controller-Skalierbarkeit mit Helm zu aktivieren, setzen Sie den `enableConcurrency` Wert auf `true`.

Neuinstallation

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Vorhandene Installation

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Alternativ setzen Sie `enableConcurrency` auf `true` in einer benutzerdefinierten `values.yaml` Datei:

```
# values.yaml
enableConcurrency: true
```

Anschließend mit der Wertedatei installieren oder aktualisieren:

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

tridentctl-Bereitstellung

Um die Skalierbarkeit des Controllers mit `tridentctl` zu ermöglichen, übergeben Sie das `--enable-concurrency` Flag während der Installation.

Neuinstallation

```
tridentctl install -n trident --enable-concurrency
```

Vorhandene Installation

Um die Controller Skalierbarkeit in einer bestehenden `tridentctl` Bereitstellung zu aktivieren, muss Trident deinstalliert und anschließend mit dem Flag erneut installiert werden:

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Überprüfen, ob die Controller Skalierbarkeit aktiviert ist

Nachdem Sie die Controller Skalierbarkeit aktiviert haben, sollte überprüft werden, ob der Trident Controller mit aktivierter Parallelverarbeitung ausgeführt wird. Die Pod-Logs des Controllers können überprüft werden:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Die Ausgabe enthält einen Protokolleintrag, der bestätigt, dass die Parallelverarbeitung aktiviert ist.

Parallelitätsverhalten

Wenn die Controller-Skalierbarkeit aktiviert ist, wendet der Trident-Controller das folgende Verhalten an:

- Trident ersetzt die einzelne globale Sperre durch eine feingranulare, ressourcenbezogene Sperrung.
- Trident serialisiert Operationen, die dieselbe Ressource verändern, um die Datenkonsistenz zu gewährleisten.
- Operationen, die nur von einer Ressource lesen, laufen gleichzeitig mit anderen Leseoperationen auf dieser Ressource ab.
- Trident begrenzt gleichzeitige ONTAP-API-Anfragen auf 20 pro Management-LIF, um eine Überlastung der Backend-Speichersysteme zu verhindern.
- Wenn mehrere Backends dieselbe Management-LIF verwenden, teilen sie sich dieses Limit von 20 Anfragen.

Vorbehalte und Einschränkungen

Für die Skalierbarkeit von Controllern in Trident 26.06 gelten folgende Überlegungen:

- Die Controller Skalierbarkeit unterstützt nur die Treiber, die in den Listen für allgemeine Verfügbarkeit und technische Vorschau aufgeführt sind. Einzelheiten enthält [Bevor Sie aktivieren](#).
- Der Trident Controller verwaltet die Parallelverarbeitung intern. Diese Version bietet keine vom Benutzer konfigurierbaren Parallelitätsgrenzen.
- Der Gesamtdurchsatz hängt vom verwendeten Speichertreiber, der Reaktionsfähigkeit des Backends und der Leistung des Kubernetes API-Servers ab.
- Hohe Parallelität kann die Belastung der Backend-Speichersysteme erhöhen.
- Das Skalierbarkeitsverhalten des Controllers ist nicht bei allen Treibern identisch.
- Der Treiber in der technischen Vorschau kann unter hoher Last eine inkonsistente Performance aufweisen und das Verhalten kann sich zwischen den Versionen ändern.
- Die Fehlersuche bei gleichzeitig laufenden Operationen kann aufgrund der parallelen Ausführung komplexer sein. Metriken und Protokolle können verschachtelte Operationsausgaben anzeigen.

Empfehlungen

Bevor Sie die Controller Skalierbarkeit aktivieren, sind die folgenden Schritte abzuschließen:

1. Bestätigt werden muss, dass jedes konfigurierte Backend einen Treiber aus der Liste der allgemein verfügbaren oder technischen Vorschau-Treiber verwendet.
2. Die Änderung sollte in einem Nicht-Produktionscluster getestet werden, bevor sie in der Produktionsumgebung angewendet wird.
3. Es sollte sichergestellt sein, dass die Controller Skalierbarkeit nach der Anwendung der Änderung aktiviert ist.

Die folgenden allgemeinen Empfehlungen sind beim Betrieb mit Controller-Skalierbarkeit zu berücksichtigen:

- Allgemein verfügbare Treiber eignen sich für Produktionsumgebungen, die eine hohe Skalierbarkeit erfordern.
- Der technische Vorschau-Treiber ist ausschließlich in Nicht-Produktionsumgebungen zu evaluieren.
- Die Backend- und Controller-Performance bei großem Maßstab im Blick behalten.

- In Automatisierungsskripten sollte nicht von einer bestimmten Reihenfolge der Vorgänge ausgegangen werden.

Automatische Volumenerweiterung

Die automatische Volumenerweiterung ermöglicht es, dass von Trident bereitgestellte Persistent Volumes automatisch wachsen, sobald die genutzte Kapazität einen definierten Schwellenwert erreicht. Diese Funktion reduziert den Betriebsaufwand und trägt dazu bei, Anwendungsunterbrechungen aufgrund von Kapazitätsengpässen zu vermeiden. Die automatische Volumenerweiterung wird mithilfe von Autogrow Policies implementiert. Eine Autogrow Policy definiert:

- Die Auslastungsschwelle, die die Erweiterung auslöst
- Die Menge, um die das Volumen wächst
- Die maximale Größe, die das Volume erreichen kann



Volumes erhöhen sich automatisch, wenn der festgelegte Auslastungsschwellenwert überschritten wird. Volumes werden niemals automatisch reduziert.

Anforderungen

Bevor Sie die automatische Volumenerweiterung konfigurieren, stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Trident 26.02 oder später
- Rollenbasierte Zugriffssteuerungsberechtigungen zum Erstellen `TridentAutogrowPolicy` benutzerdefinierter Ressourcen
- StorageClasses konfiguriert mit `allowVolumeExpansion: true`
- Unterstützte ONTAP-Protokolle:
 - Netzwerkdateisystem (NFS)
 - Internet Small Computer Systems Interface (iSCSI)
 - Non-Volatile Memory Express (NVMe)
 - Fibre Channel Protocol (FCP)

Einschränkungen

- ONTAP Non-Volatile Memory Express Raw-Block-Volumes früher als ONTAP 9.16.1 unterstützen keine automatische Erweiterung.
- Für SAN-Volumes, wenn `growthAmount` kleiner oder gleich 50 Mebibyte ist, erhöht Trident den Wert automatisch auf 51 Mebibyte, bevor die Größe angepasst wird, vorausgesetzt, die resultierende Größe überschreitet nicht `maxSize`.
- In Brownfield-Umgebungen funktioniert die automatische Erweiterung möglicherweise nicht für bestimmte vorhandene Volumes aufgrund des Migrationsverhaltens bei der Volume-Veröffentlichung.
- Wenn ein Volumen `maxSize` erreicht, findet keine weitere Erweiterung statt.

- Unterstützte Protokolle für die automatische Volumenerweiterung:
 - Netzwerkdateisystem (NFS)
 - Internet Small Computer Systems Interface (iSCSI)
 - Non-Volatile Memory Express (NVMe)
 - Fibre Channel Protocol (FCP)

Volumes mit Autogrow-Richtlinie bereitstellen

Die Autogrow Policy kann auf zwei Ebenen konfiguriert werden:

- Speicherebene: Legt den Standardwert für alle Volumes fest (mittels Annotation)
- PVC-Ebene: Überschreibt die Standardeinstellung der Speicherklasse (using annotation)

Erstellen Sie eine Autogrow-Richtlinie

Autogrow-Richtlinien ermöglichen die automatische Volumenerweiterung, wenn Volumes einen definierten Kapazitätsschwellenwert erreichen.

Stellen Sie sicher, dass Sie Folgendes haben:

- Trident 26.02 oder höher installiert
- Rollenbasierte Zugriffssteuerungsberechtigungen zum Erstellen `TridentAutogrowPolicy` von Ressourcen
- Verständnis der Anforderungen an das Arbeitslastwachstum

Eine Autogrow Policy definiert, wie sich Volumes automatisch vergrößern, wenn sie einen festgelegten Kapazitätsschwellenwert erreichen.

Sie können Autogrow Policies an jeder beliebigen Stelle in Ihrem Workflow erstellen:

- Bevor StorageClasses und Volumes erstellt werden
- Nachdem StorageClasses existieren
- Nachdem die Volumes bereitgestellt wurden

Diese Flexibilität ermöglicht es Ihnen, eine automatische Erweiterung einzuführen, ohne bestehende Ressourcen neu zu erstellen.

Spezifikationen der Autogrow Policy

Autogrow Policies sind benutzerdefinierte Kubernetes-Ressourcen, die wie folgt definiert sind:

Feld	Beschreibung	Format	Erforderlich	Beispiel	Standard
Name	Eindeutige Richtlinienkennung	Zeichenkette	Ja	Produktionsdatenbankrichtlinie	Keine
usedThreshold	Kapazitätsprozentsatz, der die Erweiterung auslöst	Prozentzeichenkette	Ja	"80%"	Keine

Feld	Beschreibung	Format	Erforderlich	Beispiel	Standard
growthAmount	Betrag, um den erhöht wird, wenn der Schwellenwert erreicht ist	Prozentualer Anteil oder Größe	Nein	"10%" oder "5Gi"	"10%"
maxSize	Maximale Volumengrößenbeschränkung	Kubernetes-Menge	Nein	"500Gi"	Unbegrenzt

Erstellen Sie eine Autogrow-Richtlinie

Schritte

1. Erstellen Sie eine YAML-Datei, die Ihre Autogrow Policy definiert:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Wenden Sie die Richtlinie auf Ihren Cluster an:

```
kubectl apply -f autogrow-policy.yaml
```

3. Überprüfen Sie, ob die Richtlinie erstellt wurde:

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Erwartete Ausgabe

NAME	USED THRESHOLD	GROWTH AMOUNT	STATE
standard-autogrow	80%	10%	Success

Richtlinienstatus

Nachdem Sie eine Richtlinie erstellt haben, validiert Trident die Spezifikation und weist einen der folgenden Zustände zu:

Status	Beschreibung	Aktion erforderlich
Erfolg	Die Richtlinie ist validiert und einsatzbereit.	Keine.

Status	Beschreibung	Aktion erforderlich
Fehlgeschlagen	Validierungsfehler festgestellt.	Spezifikation prüfen und korrigieren.
Löschen	Löschung wird durchgeführt.	Warten Sie auf den Abschluss.

Ordnen Sie eine Richtlinie einer StorageClass zu

Sie können eine Autogrow-Richtlinie mit einer StorageClass durch die `trident.netapp.io/autogrowPolicy` Annotation verknüpfen. Alle Volumes, die aus dieser StorageClass bereitgestellt werden, erben die Richtlinie.



Die StorageClass muss `allowVolumeExpansion: true` haben.

Schritte

1. Erstellen oder ändern Sie eine StorageClass mit der Autogrow Policy-Annotation:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Wenden Sie die StorageClass an:

```
kubectl apply -f storageclass.yaml
```

3. Überprüfen Sie die Annotation:

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Erwartete Ausgabe

```
production-db-policy
```

Richtlinienvorrang

Wenn Autogrow-Richtlinien-Annotationen sowohl für eine StorageClass als auch für ein PVC festgelegt sind, wendet Trident die folgenden Vorrangregeln an:

1. **PVC-Annotationen haben Vorrang.** Wenn eine PVC `trident.netapp.io/autogrowPolicy` festlegt, wird dieser Wert immer verwendet.
2. **StorageClass-Anmerkung gilt nur, wenn die PVC keine Anmerkung enthält.**
3. **Wenn keines der beiden die Annotation hat, wird keine Autogrow Policy angewendet.**

StorageClass Anmerkung	PVC-Anmerkung	Effektives Verhalten
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	Nicht festgelegt	Anwendungsgebiete <code>standard-agp</code> .
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	<code>trident.netapp.io/autogrowPolicy: logs-policy</code>	Verwendet <code>logs-policy</code> (PVC überschreibt StorageClass).
<code>trident.netapp.io/autogrowPolicy: standard-agp</code>	<code>trident.netapp.io/autogrowPolicy: "none"</code>	Keine Autogrow-Richtlinie (PVC deaktiviert autogrow).
Nicht festgelegt	<code>trident.netapp.io/autogrowPolicy: dev-policy</code>	Anwendungsgebiete <code>dev-policy</code> .
Nicht festgelegt	Nicht festgelegt	Keine Autogrow Policy.

Konfigurationsbeispiele

Die folgenden Beispiele zeigen gängige Autogrow Policy-Konfigurationen für verschiedene Anwendungsfälle.

Konservative Richtlinie für Produktionsdatenbanken

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Protokollspeicherung mit festen Wachstumsschritten

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
```

```
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Minimalrichtlinie mit Standardeinstellungen

Wenn Sie `growthAmount` und `maxSize` weglassen, verwendet Trident die Standardeinstellungen (10% Wachstum, unbegrenzte Größe):

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Richtlinie mit benutzerdefiniertem maxSize und standardmäßigem growthAmount

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Aggressives Wachstum mit unbegrenztem maxSize

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Richtlinie mit Bruchteilprozenten

```
apiVersion: trident.netapp.io/v1
```

```
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```



Bruchprozentsätze werden unterstützt. Wenn Sie mehr als drei Dezimalstellen angeben, rundet Trident den Wert auf drei Dezimalstellen.

NAS StorageClass mit Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass mit Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Autogrow-Richtlinien verwalten

Nachdem Sie Autogrow Policies erstellt haben, können Sie diese anzeigen, aktualisieren und löschen. Es besteht außerdem die Möglichkeit zu überwachen, welche Volumes eine

bestimmte Policy verwenden.

Autogrow-Richtlinien anzeigen

Alle Richtlinien auflisten

Verwenden Sie `kubectl`, um alle Autogrow Policies in Ihrem Cluster aufzulisten:

```
kubectl get tridentautogrowpolicy
```

Alternativ verwenden Sie `tridentctl`:

```
tridentctl get autogrowpolicy
```

Richtliniendetails anzeigen

Um die vollständige Spezifikation und den Status einer Richtlinie anzuzeigen:

```
kubectl describe tridentautogrowpolicy production-db-policy
```

So zeigen Sie eine Richtlinie mit den zugehörigen Volumes im YAML-Format an:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Eine Autogrow-Richtlinie aktualisieren

Sie können eine bestehende Richtlinie anpassen, um deren Schwellenwert, Wachstumsbetrag oder maximale Größe zu ändern. Änderungen werden sofort für alle Volumes wirksam, die diese Richtlinie verwenden.



Änderungen betreffen alle Volumes, die diese Richtlinie verwenden. Änderungen sollten nach Möglichkeit zunächst in einer Nicht-Produktionsumgebung getestet werden.

Schritte

1. Richtlinie bearbeiten:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

2. Ändern Sie die `spec` Felder nach Bedarf:

```
spec:
  usedThreshold: "75%"    # Changed from 80%
  growthAmount:  "20%"    # Changed from 10%
```

```
maxSize: "1Ti" # Changed from 500Gi
```

3. Speichern und beenden. Die Änderungen werden sofort wirksam.

Aktualisierungsüberlegungen

- **Sofortige Wirkung:** Alle Volumina, die die Richtlinie anwenden, übernehmen die neuen Parameter bei der nächsten Wachstumsbewertung.
- **Kein Neustart des Volumes erforderlich:** Änderungen gelten für den nächsten Wachstumsvorgang.
- **Erst testen:** Validieren Sie Änderungen nach Möglichkeit in einer Nicht-Produktionsumgebung.
- **Änderungen kommunizieren:** Benachrichtigen Sie die Teams, wenn Sie gemeinsam genutzte Richtlinien ändern.

Löschen einer Autogrow-Richtlinie

Autogrow Policies verwenden Finalizer-Schutz, um eine versehentliche Löschung zu verhindern, während Volumes sie aktiv verwenden.

Schritte

1. Löschen Sie die Richtlinie:

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Wenn die Volumes die Richtlinie noch verwenden, wechselt der Löschvorgang in einen `Deleting` Status. Es kann geprüft werden, welche Volumes betroffen sind:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Entfernen Sie die Richtlinie von jedem betroffenen Volume. Wählen Sie eine der folgenden Optionen:

- **Option A: Automatisches Wachstum explizit deaktivieren**, indem Sie die Annotation auf "none" setzen:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

- **Option B: Die Annotation vollständig entfernen:**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Löschverhalten

Szenario	Verhalten
Keine Volumes verwenden die Richtlinie	Die Richtlinie wird sofort gelöscht.
Volumes verwenden die Richtlinie	Die Richtlinie tritt in <code>Deleting</code> Zustand ein. Ein Finalisierer blockiert den Abschluss, bis alle Volumes entfernt sind.
Alle Volumes werden aus der Richtlinie entfernt	Finalisierer werden entfernt und die Richtlinie wird gelöscht.

Autogrow-Richtliniennutzung überwachen

Überprüfen Sie Volumes mithilfe einer Richtlinie

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Ermitteln Sie, welche Richtlinie ein Volume verwendet

```
kubectl get pvc database-pvc -o  
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Richtlinienereignisse überwachen

```
kubectl get events --field-selector  
involvedObject.kind=TridentAutogrowPolicy
```

Unterstützte Protokolle

Autogrow unterstützt die folgenden Speicherprotokolle:

- NFS
- iSCSI
- FCP
- NVMe



Für SAN-Volumes, wenn die konfigurierte `growthAmount` 50 MiB oder weniger beträgt, erhöht Trident den Wachstumsbetrag für die Größenänderung automatisch auf 51 MB, solange die resultierende Größe `maxSize` nicht überschreitet.

Unterstützte Backends

Autogrow unterstützt ONTAP Backends. Autogrow unterstützt außerdem Google Cloud NetApp Volumes (GCNV) sowohl für NAS- als auch für SAN Volumes.

Bekannte Einschränkungen

- **ONTAP NVMe raw block volumes:** Volumes, die mit ONTAP Versionen vor 9.16.1 erstellt wurden, unterstützen kein automatisches Wachstum.
- **Vorhandene Volumes (Brownfield-Bereitstellungen):** Die automatische Vergrößerung funktioniert möglicherweise nicht für vorhandene Volumes, selbst wenn eine gültige Autogrow-Richtlinie angewendet wird. Diese Einschränkung ist auf eine laufende Migration der Volume-Veröffentlichungen zurückzuführen. Zur Bestätigung, dass die Migration abgeschlossen ist, sind die Trident Controller-Protokolle auf "Migration completed" Meldungen zu prüfen.

Häufig gestellte Fragen

Wann wertet Trident den Schwellenwert aus?

Trident überwacht kontinuierlich die Speicherauslastung. Wenn die belegte Kapazität die `usedThreshold` überschreitet, erstellt Trident eine interne Größenänderungsanforderung und erweitert das Volume um die konfigurierte `growthAmount`.

Diese Richtlinie löst beispielsweise eine Erweiterung bei 80 % Kapazität aus und erhöht das Volumen jedes Mal um 10 %, bis zu einem Maximum von 500 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Kann ich eine Richtlinie anwenden, nachdem Volumes bereits bereitgestellt wurden?

Ja. Sie können jederzeit eine Autogrow-Richtlinie erstellen und diese auf bestehende PVCs anwenden, indem Sie die `trident.netapp.io/autogrowPolicy` Annotation hinzufügen oder aktualisieren. Sie müssen weder die PVC noch die StorageClass neu erstellen.

So wenden Sie eine Richtlinie auf ein bestehendes PVC an:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Um eine Richtlinie auf eine bestehende StorageClass anzuwenden:

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
```

```
--overwrite
```

Was passiert, wenn ich sowohl für die StorageClass als auch für das PVC eine Autogrow-Richtlinie einrichte?

Die PVC-Annotation hat immer Vorrang. Wenn eine PVC die `trident.netapp.io/autogrowPolicy` Annotation enthält, verwendet Trident diesen Wert unabhängig davon, was die StorageClass angibt. Details sind in ["Richtlinienvorrang"](#) zu finden.

Zum Beispiel, gegeben diese StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

Und dieses PVC, das die StorageClass-Richtlinie außer Kraft setzt:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident verwendet `logs-policy` für `database-pvc`, nicht `standard-agp`.

Wie deaktiviere ich Autogrow für ein bestimmtes Volume?

Die PVC-Annotation wird auf `"none"` gesetzt. Diese Einstellung überschreibt jede Richtlinie auf StorageClass-Ebene für dieses Volume:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
```

```
--overwrite
```

Sie können überprüfen, ob autogrow deaktiviert ist:

```
kubectl get pvc <pvc-name> -o jsonpath  
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Erwartete Ausgabe

```
none
```

Was passiert, wenn ein Volumen maxSize erreicht?

Trident stellt die Erweiterung des Volumes ein. Es werden keine weiteren Größenänderungsanforderungen für dieses Volume erstellt, selbst wenn die Nutzung über den `usedThreshold` hinaus weiter zunimmt.

Beispielsweise wird mit dieser Richtlinie das Wachstum des Volumens durch Trident gestoppt, sobald es 100 GiB erreicht:

```
apiVersion: trident.netapp.io/v1  
kind: TridentAutogrowPolicy  
metadata:  
  name: capped-policy  
spec:  
  usedThreshold: "90%"  
  growthAmount: "10Gi"  
  maxSize: "100Gi"
```

Um unbegrenztes Wachstum zu ermöglichen, lassen Sie `maxSize` weg oder setzen Sie es auf 0:

```
apiVersion: trident.netapp.io/v1  
kind: TridentAutogrowPolicy  
metadata:  
  name: unlimited-policy  
spec:  
  usedThreshold: "85%"  
  growthAmount: "10%"
```

Kann ich eine Richtlinie ändern, ohne die Volumes neu zu starten?

Ja. Wenn eine Richtlinie aktualisiert wird, übernehmen alle Volumes, die diese Richtlinie verwenden, die neuen Parameter bei der nächsten Wachstumsbewertung. Ein Neustart der Volumes ist nicht erforderlich.

Um eine Richtlinie zu aktualisieren:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Passen Sie die Felder nach Bedarf an:

```
spec:
  usedThreshold: "75%"      # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

Speichern und beenden. Überprüfen Sie die aktualisierte Richtlinie:

```
kubectl get tridentautogrowpolicy production-db-policy
```

Erwartete Ausgabe

NAME	USED THRESHOLD	GROWTH AMOUNT	STATE
production-db-policy	75%	20%	Success

Warum befindet sich meine Richtlinie im Status „Fehlgeschlagen“?

Ein Failed Status zeigt an, dass die Richtlinienpezifikation Validierungsfehler enthält. Führen Sie den folgenden Befehl aus, um die Fehlerdetails anzuzeigen:

```
kubectl describe tridentautogrowpolicy <policy-name>
```

Häufige Ursachen sind ein ungültiger `usedThreshold` (muss zwischen 1–99 % liegen), ein `growthAmount`, der `maxSize` überschreitet, oder ein ungültiges Kubernetes-Mengenformat. Korrigieren Sie die Spezifikation und wenden Sie sie erneut an.

```
kubectl apply -f autogrow-policy.yaml
```

Warum kann ich eine Richtlinie nicht löschen?

Richtlinien verwenden einen Finalisierungsschutz. Wenn Volumes die Richtlinie noch verwenden, wechselt der Löschvorgang in einen `Deleting` Status und wartet, bis alle Volumes aus der Richtlinie entfernt wurden.

Identifizieren Sie die betroffenen Volumes:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Entfernen Sie anschließend die Annotation von jedem PVC:

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Nachdem alle Volumes entfernt wurden, gibt Trident den Finalizer frei und löscht die Richtlinie.

Welche Backends unterstützt Autogrow?

Autogrow unterstützt ONTAP Backends. Unterstützt werden außerdem Google Cloud NetApp Volumes (GCNV) sowohl für NAS- als auch für SAN-Volumes.

Autogrow unterstützt die Protokolle NFS, iSCSI, FCP und NVMe. NVMe Raw-Block-Volumes erfordern ONTAP 9.16.1 oder höher.

Bei bestehenden Volumes in Brownfield-Umgebungen muss die Migration der Volume-Veröffentlichung möglicherweise abgeschlossen sein, bevor das automatische Wachstum wirksam wird. Zum Überprüfen des Migrationsstatus sind die Trident Controller-Protokolle zu prüfen:

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

Die folgenden StorageClass-Beispiele zeigen Autogrow für NAS- und SAN-Backends konfiguriert:

NAS-Backend

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN-Backend

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
```

```

metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true

```

Was ist der minimale Zuwachs für SAN-Volumes?

Für SAN-Volumes beträgt die effektive Mindestwachstumsgröße 51 MB. Wenn Sie eine `growthAmount` von 50 MiB oder weniger konfigurieren, erhöht Trident das Wachstum für die Größenänderung automatisch auf 51 MB.

Diese Richtlinie legt beispielsweise ein `growthAmount` von "40Mi" fest, aber Trident wendet ein Wachstum von 51 MB für jedes SAN-Volume an, das sie verwendet:

```

apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"

```

Um diese automatische Anpassung zu vermeiden, setzen Sie `growthAmount` auf einen Wert größer als 50 MiB:

```

apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"

```

Importieren von Trident Volumes

Sie können vorhandene Speichervolumes als Kubernetes PV mit `tridentctl import` importieren oder indem Sie einen Persistent Volume Claim (PVC) mit Trident-

Importannotationen erstellen.

Überblick und Überlegungen

Sie könnten ein Volume in Trident importieren, um:

- Eine Anwendung containerisieren und ihren vorhandenen Datensatz wiederverwenden
- Verwenden Sie einen Klon eines Datensatzes für eine kurzlebige Anwendung
- Einen ausgefallenen Kubernetes-Cluster wiederherstellen
- Anwendungsdaten während der Notfallwiederherstellung migrieren

Überlegungen

Bevor Sie ein Volume importieren, beachten Sie die folgenden Hinweise.

- Trident kann nur ONTAP-Volumes vom Typ RW (Read-Write) importieren. Volumes vom Typ DP (data protection) sind SnapMirror Ziel-Volumes. Sie sollten die Spiegelungsbeziehung aufheben, bevor Sie das Volume in Trident importieren.
- Wir empfehlen, Volumes ohne aktive Verbindungen zu importieren. Um ein aktiv genutztes Volume zu importieren, klonen Sie das Volume und führen Sie dann den Import durch.



Dies ist besonders wichtig für Block-Volumes, da Kubernetes die vorherige Verbindung nicht erkennen und leicht ein aktives Volume an einen Pod anhängen könnte. Dies kann zu Datenbeschädigung führen.

- Obwohl `StorageClass` auf einem PVC angegeben werden muss, verwendet Trident diesen Parameter beim Import nicht. Speicherklassen werden während der Volume-Erstellung verwendet, um anhand der Speichereigenschaften aus den verfügbaren Pools auszuwählen. Da das Volume bereits existiert, ist beim Import keine Poolauswahl erforderlich. Daher schlägt der Import nicht fehl, selbst wenn das Volume auf einem Backend oder in einem Pool existiert, der nicht der im PVC angegebenen Speicherklasse entspricht.
- Die vorhandene Volumengröße wird ermittelt und im PVC festgelegt. Nachdem das Volumen vom Speichertreiber importiert wurde, wird das PV mit einer `ClaimRef` auf das PVC erstellt.
 - Die Rückgewinnungsrichtlinie ist anfänglich auf `retain` im PV festgelegt. Nachdem Kubernetes PVC und PV erfolgreich gebunden hat, wird die Rückgewinnungsrichtlinie aktualisiert, um der Rückgewinnungsrichtlinie der Storage Class zu entsprechen.
 - Wenn die Rückgewinnungsrichtlinie der Storage Class `delete` ist, löscht Trident das Speichervolume, wenn das PV gelöscht wird.
- Standardmäßig verwaltet Trident die PVC und benennt das FlexVol Volume sowie die LUN oder den Qtree im Backend um. Es besteht die Möglichkeit, das `--no-manage` Flag zu verwenden, um ein nicht verwaltetes Volume zu importieren, und das `--no-rename` Flag, um den Volume-Namen beizubehalten.
 - `--no-manage*` - Wenn Sie das `--no-manage` Flag verwenden, führt Trident während des gesamten Lebenszyklus der Objekte keine zusätzlichen Operationen an der PVC oder PV durch. Das Speichervolume wird beim Löschen der PV nicht gelöscht und andere Operationen wie das Klonen und die Größenänderung von Volumes werden ebenfalls ignoriert.
 - `--no-rename*` - Wenn das `--no-rename` Flag verwendet wird, behält Trident beim Importieren von Volumes den vorhandenen Volume-Namen bei und verwaltet den Lebenszyklus der Volumes. Diese Option wird nur für die `ontap-nas`, `ontap-san` (einschließlich ASA r2-Systeme), `ontap-san-economy` und `ontap-nas-economy` (qtree import) Treiber unterstützt.



Diese Optionen sind nützlich, wenn Sie Kubernetes für containerisierte Workloads verwenden möchten, aber ansonsten den Lebenszyklus des Speichervolumens außerhalb von Kubernetes verwalten möchten.

- Dem PVC und PV wird eine Anmerkung hinzugefügt, die zwei Zwecken dient: Sie zeigt an, dass das Volume importiert wurde und ob der PVC und PV verwaltet werden. Diese Anmerkung sollte nicht verändert oder entfernt werden.

Importieren Sie ein Volume

Sie können ein Volume entweder mit `tridentctl import` oder durch das Erstellen einer PVC mit Trident-Importanmerkungen importieren.



Wenn Sie PVC-Anmerkungen verwenden, müssen Sie `tridentctl` nicht herunterladen oder verwenden, um das Volume zu importieren.

Verwendung von `tridentctl`

Schritte

1. Erstellen Sie eine PVC-Datei (zum Beispiel `pvc.yaml`), die zur Erstellung des PVC verwendet wird. Die PVC-Datei sollte `name`, `namespace`, `accessModes` und `storageClassName` enthalten. Optional können Sie `unixPermissions` in Ihrer PVC-Definition angeben.

Das Folgende ist ein Beispiel für eine Mindestspezifikation:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```



Geben Sie nur die erforderlichen Parameter an. Zusätzliche Parameter wie PV-Name oder Volume-Größe können dazu führen, dass der Importbefehl fehlschlägt.

2. Verwenden Sie den `tridentctl import` Befehl, um den Namen des Trident Backend, das das Volume enthält, und den Namen, der das Volume auf dem Storage eindeutig identifiziert (zum Beispiel: ONTAP FlexVol, Element Volume), anzugeben. Das `-f` Argument ist erforderlich, um den Pfad zur PVC-Datei anzugeben.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

Verwendung von PVC-Anmerkungen

Schritte

1. Erstellen Sie eine PVC-YAML-Datei (zum Beispiel `pvc.yaml`) mit den erforderlichen Trident-Importannotationen. Die PVC-Datei sollte Folgendes enthalten:
 - `name` and `namespace` in Metadaten
 - `accessModes`, `resources.requests.storage`, und `storageClassName` in Spezifikation
 - Anmerkungen:
 - `trident.netapp.io/importOriginalName`: Volumenname auf dem Backend
 - `trident.netapp.io/importBackendUUID`: Backend-UUID, wo das Volume existiert
 - `trident.netapp.io/notManaged` (*Optional*): Auf `"true"` für nicht verwaltete Volumes festlegen. Standardmäßig ist `"false"`.

Im Folgenden finden Sie eine Beispielspezifikation für den Import eines verwalteten Volumes:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
    storageClassName: <storage-class-name>
```

2. Wenden Sie die PVC YAML-Datei auf Ihren Kubernetes-Cluster an:

```
kubectl apply -f <pvc-file>.yaml
```

Trident wird das Volumen automatisch importieren und es an das PVC binden.

Beispiele

Überprüfen Sie die folgenden Beispiele für den Volumenimport für unterstützte Treiber.

ONTAP NAS und ONTAP NAS FlexGroup

Trident unterstützt den Volumenimport mithilfe der `ontap-nas` und `ontap-nas-flexgroup` Treiber.



- Der ``ontap-nas-economy`` Treiber verwaltet Speicher auf Qtree-Ebene, daher wird das eigenständige Importieren eines FlexVol Volumes nicht unterstützt. Zum Importieren von Qtrees mit dem ``ontap-nas-economy`` Treiber siehe [Qtrees importieren](#) (Tech Preview). Wenn Trident einen Qtree importiert, wird auch das übergeordnete FlexVol als Bucket importiert, das den Qtree enthält.
- Die `ontap-nas` und ``ontap-nas-flexgroup`` Treiber erlauben keine doppelten Datenträgernamen.

Jedes mit dem `ontap-nas` Treiber erstellte Volume ist ein FlexVol Volume auf dem ONTAP Cluster. Das Importieren von FlexVol Volumes mit dem `ontap-nas` Treiber funktioniert genauso. Ein FlexVol Volume, das bereits auf einem ONTAP Cluster existiert, kann als `ontap-nas` PVC importiert werden. Ebenso können FlexGroup Volumes als `ontap-nas-flexgroup` PVCs importiert werden.

ONTAP NAS-Beispiele mit `tridentctl`

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes mit `tridentctl` importiert werden.

Verwaltetes Volume

Das folgende Beispiel importiert ein Volume mit dem Namen `managed_volume` auf einem Backend mit dem Namen `ontap_nas`:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STATE	STORAGE CLASS	MANAGED
file	pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	online	standard	true

Unverwaltetes Volume

Bei Verwendung des `--no-manage` Arguments benennt Trident das Volume nicht um.

Das folgende Beispiel importiert `unmanaged_volume` auf dem `ontap_nas` Backend:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file> --no-manage
```

```

+-----+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
+-----+-----+-----+-----+
|          BACKEND UUID  |  STATE  |  MANAGED  |
+-----+-----+-----+-----+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
+-----+-----+-----+-----+
| file          | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

ONTAP NAS-Beispiele mit PVC-Annotationen

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes mithilfe von PVC-Annotationen importiert werden.

Verwaltetes Volume

Das folgende Beispiel importiert ein 1GiB ontap-nas Volume namens `ontap_volume1` vom Backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` mit RWO-Zugriffsmodus, der mithilfe von PVC-Annotationen festgelegt ist:

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>

```

Unverwaltetes Volume

Das folgende Beispiel importiert ein 1GiB ontap-nas Volume mit dem Namen `ontap-volume2` vom Backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` mit festgelegtem RWO-Zugriffsmodus mithilfe von PVC-Annotationen:

```

kind: PersistentVolumeClaim

```

```

apiVersion: v1
metadata:
  name: <unmanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>

```

Qtrees importieren



Qtree Import ist eine Tech Preview Funktion in Trident 26.06. Tech Preview Funktionen sind für den Produktiveinsatz nicht unterstützt.

Ab Trident 26.06 ist es möglich, bestehende ONTAP qtrees mithilfe des `ontap-nas-economy` Treibers in Trident zu importieren. Der Qtree wird durch seinen FlexVol Volume- und Qtree-Namen im `flexvol/qtree` Format sowie durch die UUID des Backends, das ihn enthält, identifiziert. Die allgemeinen Importüberlegungen in [Überblick und Überlegungen](#) gelten auch für den Qtree-Import.

Einen Qtree mit `tridentctl` importieren

Geben Sie den Qtree im `flexvol/qtree` Format als Volume-Namen an. Im folgenden Beispiel wird der Qtree `qtree1` aus dem FlexVol Volume `flexvol1` auf dem Backend `ontap_nas_economy` importiert:

```
tridentctl import volume ontap_nas_economy flexvol1/qtree1 -f <path-to-pvc-file>
```

Um den Qtree zu importieren, ohne ihn umzubenennen, das `--no-rename` Flag hinzufügen. Um den Qtree zu importieren, ohne dass Trident seinen Lebenszyklus verwaltet, das `--no-manage` Flag hinzufügen.

Import eines Qtrees mithilfe von PVC-Annotationen

Eine PVC mit den folgenden Annotationen erstellen:

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>

```

```

annotations:
  trident.netapp.io/importOriginalName: "flexvoll1/qtree1"
  trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>

```

Um die bestehenden FlexVol volume- und qtree-Namen beizubehalten, die `trident.netapp.io/importNoRename: "true"` Annotation hinzufügen. Um den qtree als nicht verwaltet zu importieren, die `trident.netapp.io/notManaged: "true"` Annotation hinzufügen.

Benennungs- und Managementverhalten

Verwalteter Import (Standard)

Trident benennt den Qtree entsprechend dem PVC-Namen um und benennt das FlexVol volume gemäß der `ontap-nas-economy` Namenskonvention um. Der Verbindungspfad bleibt unverändert.

--no-rename

Trident behält die bestehenden FlexVol volume- und qtree-Namen bei. Die Namen müssen bereits den Trident Namenskonventionen entsprechen, sonst kann Trident keine Lösch- oder Mount-Operationen für den qtree verwalten.

--no-manage

Trident referenziert den qtree, verwaltet aber nicht dessen Lebenszyklus. Die Namen müssen den Trident Namenskonventionen entsprechen, und Trident verwendet oder verwaltet keine Auto-Export-Richtlinien für den qtree.

Verhalten der Export-Policy

Wenn das Backend automatische Exportrichtlinien verwendet, nutzt der importierte Qtree das Exportrichtlinienmodell pro Qtree, bei dem jeder Qtree seine eigene Exportrichtlinie besitzt. Trident weist dem Qtree beim Import eine leere Exportrichtlinie zu. Beim Veröffentlichen aktualisiert Trident das FlexVol volume auf die Exportrichtlinie pro Backend, falls der automatische Export aktiviert ist und die bestehende FlexVol Exportrichtlinie `default` oder leer ist.



Die Aktualisierung der Export-Richtlinie des FlexVol Volumes kann andere Qtrees im selben FlexVol Volume beeinträchtigen, die derzeit eingebunden sind. Qtrees sollten nur importiert werden, wenn die Qtrees im FlexVol Volume ausgehängt sind.



Zwei Kubernetes-Cluster dürfen nicht auf dieselbe aktive SVM verweisen, und es darf kein Qtree importiert werden, der noch von einem anderen Cluster verwaltet wird. Wenn beide Cluster dasselbe Volume verwalten, können widersprüchliche Vorgänge die Daten beschädigen oder zerstören.

ONTAP SAN

Trident unterstützt den Volume-Import mit den `ontap-san` (iSCSI, NVMe/TCP und FC) und `ontap-san-`

economy-Treibern.

Trident kann ONTAP SAN FlexVol Volumes importieren, die eine einzelne LUN enthalten. Dies entspricht dem `ontap-san` Treiber, der für jede PVC ein FlexVol Volume erstellt und eine LUN innerhalb des FlexVol Volumes anlegt. Trident importiert das FlexVol Volume und ordnet es der PVC-Definition zu. Trident kann `ontap-san-economy` Volumes importieren, die mehrere LUNs enthalten.

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes importiert werden:

Verwaltetes Volume

Für verwaltete Volumes benennt Trident das FlexVol Volume in das `pvc-<uuid>` Format um und die LUN innerhalb des FlexVol Volume in `lun0` um.

Das folgende Beispiel importiert das `ontap-san-managed` FlexVol Volume, das sich auf dem `ontap_san_default` Backend befindet:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d
```

NAME	SIZE	STORAGE CLASS
pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a	20 MiB	basic
cd394786-ddd5-4470-adc3-10c5ce4ca757	online	true

Unverwaltetes Volume

Das folgende Beispiel importiert `unmanaged_example_volume` auf dem `ontap_san` Backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
```

NAME	SIZE	STORAGE CLASS
pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228	1.0 GiB	san-blog
e3275890-7d80-4af6-90cc-c7a0759f555a	online	false

Wenn Sie LUNs Igroups zugeordnet haben, die denselben IQN wie ein Kubernetes-Knoten-IQN verwenden (siehe folgendes Beispiel), erhalten Sie die Fehlermeldung: `LUN already mapped to initiator(s) in this group`. Sie müssen den Initiator entfernen oder die Zuordnung der LUN aufheben, um das Volume zu importieren.

Vserver	Igroup	Protocol	OS Type	Initiators
svm0	k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3	iscsi	linux	iqn.1994-05.com.redhat:4c2e1cf35e0
svm0	unmanaged-example-igroup	mixed	linux	iqn.1994-05.com.redhat:4c2e1cf35e0

ONTAP SAN-economy Beispiele

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes für das `ontap-san-economy` Backend importiert werden.

Verwaltetes Volume

Wenn Sie ein verwaltetes Volume importieren, übernimmt Trident die Eigentümerschaft des FlexVol und benennt es um. Sie müssen diese Umbenennung berücksichtigen, wenn Sie mehrere LUNs aus demselben FlexVol importieren.

Das folgende Beispiel importiert `lun1` aus der FlexVol `toimport` als verwaltetes Volume mit dem Namen `vol-managed-saneco`:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Nach dem Import `lun1` benennt Trident die FlexVol (zum Beispiel in `trident_lun_pool_xyz`) um. Um zusätzliche LUNs aus derselben FlexVol zu importieren, verwenden Sie den neuen FlexVol-Namen:

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```



Das `ontap-san-economy` Backend importiert jeweils eine LUN. Sie können mehrere Importe mithilfe eines Skripts automatisieren.

Unverwaltetes Volume

Wenn Sie ein nicht verwaltetes Volume importieren, übernimmt Trident nicht die Besitzrechte an dem FlexVol. Allerdings müssen das FlexVol und die LUN den Trident-Namenskonventionen entsprechen.

FlexVol-Benennungsformat

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- STORAGEPREFIX ist der Wert von `storagePrefix` in Ihrer Backend-Konfiguration. Der Standardwert ist `trident`.
- RANDOMSTRING ist eine beliebige Zeichenkette Ihrer Wahl.

LUN-Benennungsanforderung

Der LUN muss mit einem Namen versehen werden `lun0`.

Beispiel

Wenn Ihre `storagePrefix` `xyz` ist, lautet der vollständige Pfad zur LUN:

```
trident_lun_pool_xyz_randomstring/lun0
```

Element

Trident unterstützt NetApp Element Software und NetApp HCI Volume-Import mit dem `solidfire-san` Treiber.



Der Element-Treiber unterstützt doppelte Volume-Namen. Trident gibt jedoch einen Fehler zurück, wenn doppelte Volume-Namen vorhanden sind. Als Workaround klonen Sie das Volume, vergeben einen eindeutigen Volume-Namen und importieren das geklonte Volume.

Das folgende Beispiel importiert ein `element-managed` Volume auf Backend `element_default`.

```
tridentctl import volume element_default element-managed -f pvc-basic-import.yaml -n trident -d
```

	NAME		SIZE		STORAGE CLASS
PROTOCOL		BACKEND UUID		STATE	
	pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe		10 GiB		basic-element
block		d3ba047a-ea0b-43f9-9c42-e38e58301c49		online	


```
|
+-----+
+-----+-----+-----+
+-----+-----+
```

Das folgende Beispiel importiert ein `google-cloud-netapp-volumes` Volume, wenn sich zwei Volumes in derselben Region befinden:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident
```

	NAME	SIZE	STORAGE CLASS
PROTOCOL	BACKEND UUID	STATE	MANAGED
pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0	10 GiB	gcnv-nfs-sc-	
identity file	8c18cdf1-0770-4bc0-bcc5-c6295fe6d837	online	true

Anpassung von Trident Volume-Namen und Labels

Mit Trident können Sie den von Ihnen erstellten Volumes aussagekräftige Namen und Bezeichnungen zuweisen. Dies hilft Ihnen, Volumes zu identifizieren und sie einfach ihren jeweiligen Kubernetes-Ressourcen (PVCs) zuzuordnen. Sie können außerdem auf Backend-Ebene Vorlagen für benutzerdefinierte Volume-Namen und benutzerdefinierte Bezeichnungen definieren; alle Volumes, die Sie erstellen, importieren oder klonen, halten sich an diese Vorlagen.

Bevor Sie beginnen

Unterstützung für anpassbare Volume-Namen und Labels:

- Volume-Erstellungs-, Import- und Klonvorgänge.
- Im Fall des `ontap-nas-economy`-Treibers entspricht nur der Name des Qtree-Volumes der Namensvorlage.
- Im Falle des `ontap-san-economy` Treibers entspricht nur der LUN-Name der Namensvorlage.

Einschränkungen

- Benutzerdefinierte Volumennamen sind nur mit ONTAP-Treibern für lokale Installationen kompatibel.
- Benutzerdefinierte Bezeichnungen werden nur für die `ontap-san`, `ontap-nas` und `ontap-nas-flexgroup` Treiber unterstützt.
- Benutzerdefinierte Volumennamen gelten nicht für vorhandene Volumen.

Wichtige Verhaltensweisen von anpassbaren Volumennamen

- Tritt aufgrund ungültiger Syntax in einer Namensvorlage ein Fehler auf, schlägt die Backend-Erstellung fehl. Schlägt jedoch die Vorlagenanwendung fehl, wird das Volume gemäß der bestehenden Namenskonvention benannt.
- Das Speicherpräfix ist nicht anwendbar, wenn ein Volume mithilfe einer Namensvorlage aus der Backend-Konfiguration benannt wird. Jeder gewünschte Präfixwert kann direkt zur Vorlage hinzugefügt werden.

Backend-Konfigurationsbeispiele mit Namensvorlage und Labels

Benutzerdefinierte Namensvorlagen können auf Stamm- und/oder Poolebene definiert werden.

Beispiel auf Wurzelebene

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
      "{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.RequestName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Beispiel auf Poolebene

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
```

```

"svm": "svm0",
"username": "<admin>",
"password": "<password>",
"useREST": true,
"storage": [
  {
    "labels": {
      "labelname": "label1",
      "name": "{{ .volume.Name }}"
    },
    "defaults": {
      "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
    }
  },
  {
    "labels": {
      "cluster": "label2",
      "name": "{{ .volume.Name }}"
    },
    "defaults": {
      "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster }}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
    }
  }
]
}

```

Beispiele für Namensvorlagen

Beispiel 1:

```

"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{ .config.BackendName }}"

```

Beispiel 2:

```

"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{ slice .volume.RequestName 1 5 }}"

```

Zu berücksichtigende Punkte

1. Bei Datenträgerimporten werden die Bezeichnungen nur dann aktualisiert, wenn der vorhandene Datenträger Bezeichnungen in einem bestimmten Format aufweist. Beispiel:

```
{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}.
```

2. Bei verwalteten Volume-Importen folgt der Volume-Name der Namensvorlage, die auf der Root-Ebene in der Backend-Definition definiert ist.
3. Trident unterstützt die Verwendung eines Slice-Operators mit dem Speicherpräfix nicht.
4. Falls die Vorlagen keine eindeutigen Volume-Namen ergeben, fügt Trident einige zufällige Zeichen hinzu, um eindeutige Volume-Namen zu erstellen.
5. Wenn der benutzerdefinierte Name für ein NAS-Economy-Volume länger als 64 Zeichen ist, benennt Trident die Volumes gemäß der bestehenden Namenskonvention. Bei allen anderen ONTAP Treibern schlägt der Volume-Erstellungsprozess fehl, wenn der Volume-Name die Namensbeschränkung überschreitet.

Ein Trident NFS Volume über Namespaces hinweg freigeben

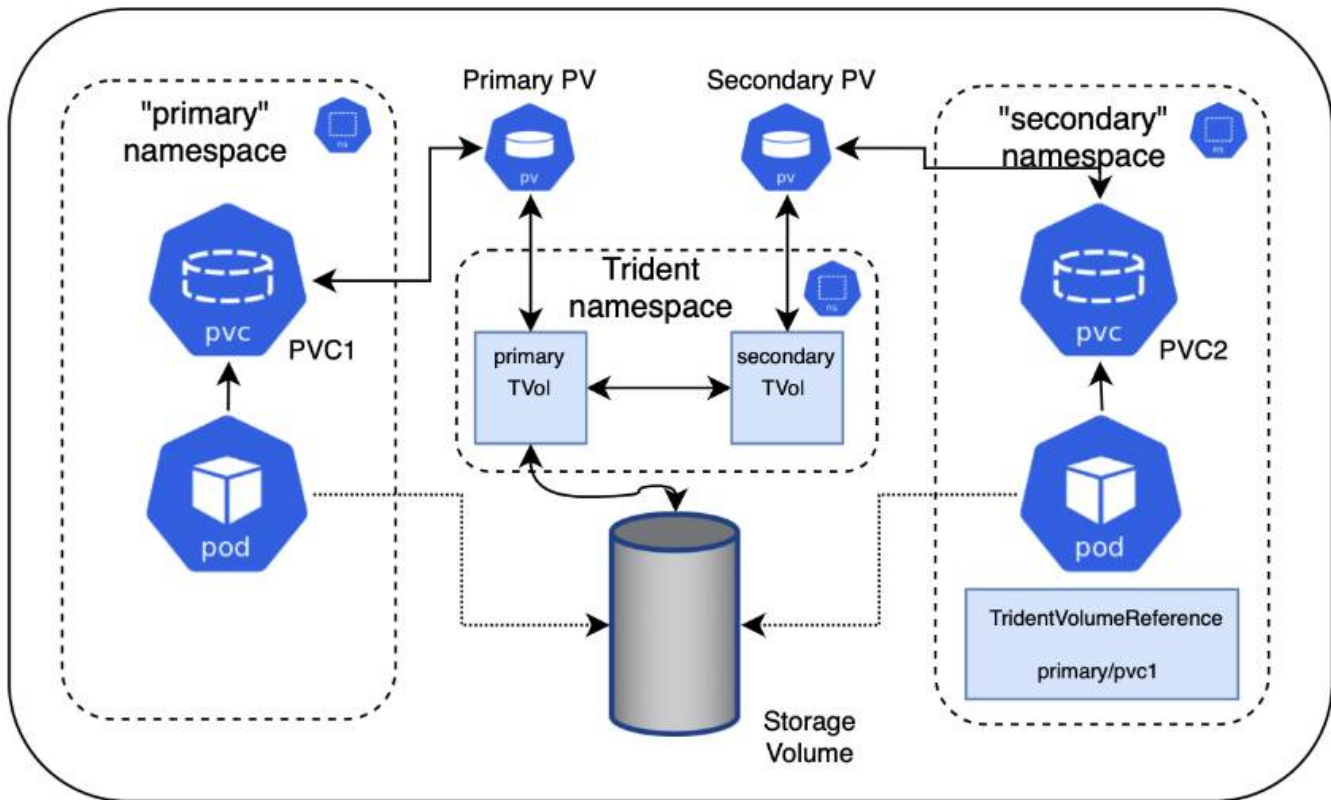
Mit Trident können Sie ein Volume in einem primären Namespace erstellen und es in einem oder mehreren sekundären Namespaces freigeben.

Features

Das TridentVolumeReference CR ermöglicht es Ihnen, ReadWriteMany (RWX)-NFS-Volumes sicher über einen oder mehrere Kubernetes-Namespaces hinweg freizugeben. Diese Kubernetes-native Lösung bietet folgende Vorteile:

- Mehrere Zugriffskontrollebenen zur Gewährleistung der Sicherheit
- Funktioniert mit allen Trident NFS-Volume-Treibern
- Keine Abhängigkeit von tridentctl oder anderen nicht-nativen Kubernetes-Funktionen

Dieses Diagramm veranschaulicht die gemeinsame Nutzung von NFS-Volumes über zwei Kubernetes-Namespaces hinweg.



Schnellstart

Sie können die NFS-Volume-Freigabe in nur wenigen Schritten einrichten.

1

Konfigurieren Sie den Quell-PVC, um das Volume freizugeben

Der Inhaber des Quell-Namespace erteilt die Berechtigung zum Zugriff auf die Daten im Quell-PVC.

2

Erteilen Sie die Berechtigung zum Erstellen eines CR im Ziel-Namespace

Der Clusteradministrator erteilt dem Eigentümer des Ziel-Namespace die Berechtigung, die TridentVolumeReference CR zu erstellen.

3

Erstellen Sie TridentVolumeReference im Ziel-Namespace

Der Eigentümer des Ziel-Namensraums erstellt die TridentVolumeReference CR, um auf die Quell-PVC zu verweisen.

4

Erstellen Sie den untergeordneten PVC im Ziel-Namespace

Der Eigentümer des Ziel-Namensraums erstellt den untergeordneten PVC, um die Datenquelle aus dem Quell-PVC zu verwenden.

Konfigurieren Sie die Quell- und Ziel-Namespace

Um die Sicherheit zu gewährleisten, erfordert die gemeinsame Nutzung über verschiedene Namespaces hinweg die Zusammenarbeit und das Handeln des Quell-Namespace-Inhabers, des Cluster-Administrators und des Ziel-Namespace-Inhabers. Die Benutzerrolle wird in jedem Schritt festgelegt.

Schritte

1. **Quellnamespace-Inhaber:** Erstellen Sie die PVC (pvc1 im Quellnamespace, die die Berechtigung zur gemeinsamen Nutzung mit dem Zielnamespace (namespace2 mithilfe der `shareToNamespace` Annotation gewährt.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident erstellt das PV und sein Backend-NFS-Speichervolume.



- Sie können die PVC mithilfe einer durch Kommas getrennten Liste für mehrere Namensräume freigeben. Zum Beispiel `trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4`.
- Sie können für alle Namensräume freigeben `*`. Zum Beispiel, `trident.netapp.io/shareToNamespace: *`
- Sie können die PVC jederzeit aktualisieren, um die `shareToNamespace` Annotation einzuschließen.

2. **Cluster-Administrator:** Stellen Sie sicher, dass die korrekte RBAC-Konfiguration vorhanden ist, um dem Besitzer des Ziel-Namespace die Berechtigung zum Erstellen des `TridentVolumeReference` CR im Ziel-Namespace zu erteilen.
3. **Ziel-Namespace-Inhaber:** Erstellen Sie eine `TridentVolumeReference` CR im Ziel-Namespace, die auf den Quell-Namespace verweist `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
```

```
namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Ziel-Namespace-Inhaber:** Erstellen Sie einen PVC (pvc2 im Ziel-Namespace (namespace2 unter Verwendung der `shareFromPVC`-Annotation, um den Quell-PVC zu kennzeichnen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```



Die Größe des Ziel-PVC muss kleiner oder gleich der des Quell-PVC sein.

Ergebnisse

Trident liest die `shareFromPVC` Annotation auf dem Ziel-PVC und erstellt das Ziel-PV als untergeordnetes Volume ohne eigene Speicherressource, das auf das Quell-PV verweist und die Speicherressource des Quell-PV mitnutzt. Das Ziel-PVC und -PV erscheinen wie gewohnt verbunden.

Löschen eines freigegebenen Volumes

Sie können ein Volume löschen, das von mehreren Namespaces gemeinsam genutzt wird. Trident entfernt den Zugriff auf das Volume im Quell-Namespace und erhält den Zugriff für andere Namespaces, die das Volume gemeinsam nutzen. Wenn alle Namespaces, die auf das Volume verweisen, entfernt wurden, löscht Trident das Volume.

Verwenden Sie `tridentctl get` zur Abfrage untergeordneter Volumes

Mit dem `[tridentctl-Dienstprogramm` kann der `get`-Befehl ausgeführt werden, um untergeordnete Volumes abzurufen. Weitere Informationen finden Sie unter `tridentctl` [commands and options](#).

```
Usage:
  tridentctl get [option]
```

Flags:

- `-h, --help`: Hilfe für Bände.
- `--parentOfSubordinate string`: Beschränken Sie die Abfrage auf das untergeordnete Quellvolume.
- `--subordinateOf string`: Beschränken Sie die Abfrage auf untergeordnete Elemente des Volumens.

Einschränkungen

- Trident kann nicht verhindern, dass Ziel-Namespace auf das gemeinsam genutzte Volume schreiben. Sie sollten Dateisperren oder andere Verfahren verwenden, um das Überschreiben von Daten auf dem gemeinsam genutzten Volume zu verhindern.
- Sie können den Zugriff auf die Quell-PVC nicht widerrufen, indem Sie die `shareToNamespace` oder `shareFromNamespace` Anmerkungen entfernen oder die `TridentVolumeReference` CR löschen. Um den Zugriff zu widerrufen, müssen Sie die untergeordnete PVC löschen.
- Snapshots, Klone und Spiegelungen sind auf untergeordneten Volumes nicht möglich.

Für weitere Informationen

Um mehr über den Volume-Zugriff über Namespaces hinweg zu erfahren:

- Sehen Sie sich die Demo auf ["NetAppTV"](#) an.

Klonen von Trident Volumes über Namespaces hinweg

Mit Trident können Sie neue Volumes erstellen, indem Sie vorhandene Volumes oder Volumesnapshots aus einem anderen Namespace innerhalb desselben Kubernetes-Clusters verwenden.

Voraussetzungen

Bevor Sie Volumes klonen, stellen Sie sicher, dass das Quell- und das Ziel-Backend vom gleichen Typ sind und die gleiche Speicherklasse haben.



Das Klonen über Namensräume hinweg wird nur für die `ontap-san` und `ontap-nas` Speichertreiber unterstützt. Schreibgeschützte Klone werden nicht unterstützt.

Schnellstart

Das Klonen von Volumes lässt sich in wenigen Schritten einrichten.

1

Konfigurieren Sie die Quell-PVC, um das Volume zu klonen

Der Inhaber des Quell-Namespace erteilt die Berechtigung zum Zugriff auf die Daten im Quell-PVC.

2

Erteilen Sie die Berechtigung zum Erstellen eines CR im Ziel-Namespace

Der Clusteradministrator erteilt dem Eigentümer des Ziel-Namespace die Berechtigung, die

TridentVolumeReference CR zu erstellen.

3

Erstellen Sie TridentVolumeReference im Ziel-Namespace

Der Eigentümer des Ziel-Namensraums erstellt die TridentVolumeReference CR, um auf die Quell-PVC zu verweisen.

4

Erstellen Sie den Klon-PVC im Ziel-Namespace

Der Eigentümer des Ziel-Namensraums erstellt einen PVC, um den PVC aus dem Quell-Namensraum zu klonen.

Konfigurieren Sie die Quell- und Ziel-Namespace

Um die Sicherheit zu gewährleisten, erfordert das Klonen von Volumes über Namespaces hinweg die Zusammenarbeit und das Eingreifen des Quell-Namespace-Inhabers, des Cluster-Administrators und des Ziel-Namespace-Inhabers. Die Benutzerrolle wird in jedem Schritt festgelegt.

Schritte

1. **Quellnamespace-Inhaber:** Erstellen Sie die PVC (pvc1 im Quellnamespace (namespace1, die die Berechtigung zur gemeinsamen Nutzung mit dem Zielnamespace (namespace2 mithilfe der cloneToNamespace-Annotation gewährt.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident erstellt das PV und dessen Backend-Speichervolume.



- Sie können die PVC mithilfe einer durch Kommas getrennten Liste für mehrere Namensräume freigeben. Zum Beispiel `trident.netapp.io/cloneToNamespace: namespace2, namespace3, namespace4`.
- Sie können für alle Namensräume freigeben `*`. Zum Beispiel, `trident.netapp.io/cloneToNamespace: *`
- Sie können die PVC jederzeit aktualisieren, um die `cloneToNamespace` Annotation

einzuschließen.

2. **Cluster-Administrator:** Stellen Sie sicher, dass die korrekte RBAC-Konfiguration vorhanden ist, um dem Besitzer des Ziel-Namespace die Berechtigung zum Erstellen des TridentVolumeReference CR im Ziel-Namespace zu erteilen (namespace2).
3. **Ziel-Namespace-Inhaber:** Erstellung einer TridentVolumeReference CR im Ziel-Namespace, die auf die Quell-PVC pvc1 verweist.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Ziel-Namespace-Inhaber:** Erstellen Sie einen PVC (pvc2) im Ziel-Namespace (namespace2) mit der Verwendung von cloneFromPVC oder cloneFromSnapshot und cloneFromNamespace Annotationen, um den Quell-PVC zu kennzeichnen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Einschränkungen

- Für PVCs, die mit ontap-nas-economy-Treibern bereitgestellt wurden, sind schreibgeschützte Klone nicht unterstützt.

Volumes replizieren mit SnapMirror

Trident unterstützt Spiegelungsbeziehungen zwischen einem Quellvolume in einem

Cluster und dem Zielvolume im verbundenen Cluster zur Datenreplikation für die Notfallwiederherstellung. Sie können eine namensbasierte Custom Resource Definition (CRD), genannt Trident Mirror Relationship (TMR), verwenden, um die folgenden Operationen durchzuführen:

- Spiegelbeziehungen zwischen Volumes (PVCs) erstellen
- Spiegelbeziehungen zwischen Volumes entfernen
- Spiegelbeziehungen aufheben
- Das sekundäre Volume während Katastrophenbedingungen (Failover) promoten
- Führen Sie einen verlustfreien Übergang von Anwendungen von Cluster zu Cluster durch (während geplanter Failover oder Migrationen)

Replikationsvoraussetzungen

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind, bevor Sie beginnen:

ONTAP Cluster

- **Trident:** Trident Version 22.10 oder höher muss auf beiden Quell- und Ziel-Kubernetes-Clustern vorhanden sein, die ONTAP als Backend nutzen.
- **Lizenzen:** ONTAP SnapMirror asynchrone Lizenzen mit dem Data Protection Bundle müssen sowohl auf dem Quell- als auch auf dem Ziel-ONTAP-Cluster aktiviert sein. Weitere Informationen finden Sie unter ["SnapMirror Lizenzierungsübersicht in ONTAP"](#).

Ab ONTAP 9.10.1 werden alle Lizenzen als NetApp Lizenzdatei (NLF) bereitgestellt, bei der es sich um eine einzelne Datei handelt, die mehrere Funktionen aktiviert. Weitere Informationen finden Sie unter ["In ONTAP One enthaltene Lizenzen"](#).



Es wird ausschließlich SnapMirror asynchroner Schutz unterstützt.

Peering

- **Cluster und SVM:** Die ONTAP-Speicher-Backends müssen per Peering verbunden sein. Weitere Informationen finden Sie unter ["Cluster- und SVM-Peering-Übersicht"](#).



Stellen Sie sicher, dass die in der Replikationsbeziehung zwischen zwei ONTAP Clustern verwendeten SVM-Namen eindeutig sind.

- **Trident und SVM:** Die verbundenen Remote-SVMs müssen für Trident auf dem Ziel-Cluster verfügbar sein.

Unterstützte Treiber

NetApp Trident unterstützt die Volume-Replikation mit NetApp SnapMirror-Technologie unter Verwendung von Storage-Klassen, die von den folgenden Treibern unterstützt werden: **ontap-nas: NFS** **ontap-san: iSCSI** **ontap-san: FC** **ontap-san: NVMe/TCP** (erfordert mindestens ONTAP Version 9.15.1)



Die Volume-Replikation mit SnapMirror wird für ASA r2-Systeme nicht unterstützt. Weitere Informationen zu ASA r2-Systemen finden Sie unter ["Erfahren Sie mehr über ASA r2-Speichersysteme"](#).

Erstellen Sie ein gespiegeltes PVC

Folgen Sie diesen Schritten und verwenden Sie die CRD-Beispiele, um eine Spiegelbeziehung zwischen primären und sekundären Volumes zu erstellen.

Schritte

1. Führen Sie die folgenden Schritte auf dem primären Kubernetes-Cluster aus:
 - a. Erstelle ein StorageClass Objekt mit dem `trident.netapp.io/replication: true` Parameter.

Beispiel

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

- b. Erstellen Sie eine PVC mit zuvor erstellter StorageClass.

Beispiel

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

- c. Erstellen Sie eine MirrorRelationship CR mit lokalen Informationen.

Beispiel

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
```

```
volumeMappings:
- localPVCName: csi-nas
```

Trident ruft die internen Informationen für das Volume und den aktuellen Datenschutzzustand (DP) des Volumes ab und füllt dann das Statusfeld des MirrorRelationship.

- d. Rufen Sie das TridentMirrorRelationship CR ab, um den internen Namen und die SVM des PVC zu erhalten.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Führen Sie die folgenden Schritte auf dem sekundären Kubernetes-Cluster aus:

- a. Erstellen Sie eine StorageClass mit dem Parameter trident.netapp.io/replication: true.

Beispiel

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

- b. Erstellen Sie ein MirrorRelationship CR mit Ziel- und Quellinformationen.

Beispiel

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  replicationPolicy: MirrorAllSnapshots
  replicationSchedule: 5min
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
      "datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Das TMR spec verfügt über vier Felder der obersten Ebene: `state`, das den gewünschten Beziehungsstatus festlegt (siehe [Volumenreplikationszustände](#)); `volumeMappings`, das lokale PVCs ihren Remote-Volume-Handles zuordnet; sowie die optionalen `replicationPolicy` und `replicationSchedule` Felder, die unten beschrieben sind.

Die `replicationPolicy` und `replicationSchedule` Felder sind optional. Als Wert für `replicationPolicy` ist der Name einer asynchronen SnapMirror Richtlinie auf ONTAP und für `replicationSchedule` der Name eines ONTAP Zeitplans anzugeben. Wenn eines der Felder ausgelassen wird, verwendet Trident die entsprechende Replikationsrichtlinie bzw. den entsprechenden Zeitplan, die bzw. der im Backend konfiguriert ist. Diese Felder sind beim Erstellen der Beziehung festzulegen; die Richtlinie oder der Zeitplan können nach dem Einrichten der Beziehung nicht mehr geändert werden. Wenn andere Einstellungen verwendet werden sollen, ist die TMR zu löschen und neu zu erstellen.

Trident erstellt und initialisiert eine SnapMirror Beziehung anhand der konfigurierten Replikationsrichtlinie und des Zeitplans.

1. Erstellen Sie eine PVC mit zuvor erstellter StorageClass, um als sekundäres (SnapMirror Ziel) zu fungieren.

Beispiel

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
```

```
storageClassName: csi-nas
```

Trident prüft, ob die TridentMirrorRelationship CRD vorhanden ist, und schlägt die Erstellung des Volumes fehl, falls die Beziehung nicht existiert. Wenn die Beziehung besteht, stellt Trident sicher, dass das neue FlexVol volume auf einer SVM platziert wird, die mit der Remote-SVM, die in der MirrorRelationship definiert ist, verbunden ist.

Volumenreplikationszustände

Eine Trident Mirror Relationship (TMR) ist eine CRD, die ein Ende einer Replikationsbeziehung zwischen PVCs darstellt. Die Ziel-TMR hat einen Zustand, der Trident den gewünschten Zustand mitteilt. Die Ziel-TMR hat die folgenden Zustände:

- **Festgestellt:** Das lokale PVC ist das Zielvolume einer SnapMirror Beziehung, und dies ist eine neue Beziehung.
- **Promoted:** Die lokale PVC ist ReadWrite und mountbar, es besteht derzeit keine Spiegelbeziehung.
- **Wiederhergestellt:** Das lokale PVC ist das Zielvolume einer SnapMirror Beziehung und war auch zuvor in dieser SnapMirror Beziehung.
 - Der wiederhergestellte Zustand muss verwendet werden, wenn das Zielvolume jemals in einer Beziehung mit dem Quellvolume stand, da er den Inhalt des Zielvolumes überschreibt.
 - Der wiederhergestellte Zustand schlägt fehl, wenn das Volume zuvor nicht in einer Beziehung mit der Quelle stand.

Sekundäres PVC während eines ungeplanten Failovers promoten

Führen Sie den folgenden Schritt auf dem sekundären Kubernetes-Cluster durch:

- Aktualisiere das Feld `spec.state` von TridentMirrorRelationship auf `promoted`.

Sekundäre PVC während eines geplanten Failovers promoten

Führen Sie während eines geplanten Failovers (Migration) die folgenden Schritte aus, um den sekundären PVC zu promoten:

Schritte

1. Erstellen Sie im primären Kubernetes-Cluster einen Snapshot des PVC und warten Sie, bis der Snapshot erstellt wurde.
2. Im primären Kubernetes-Cluster wird der TridentTridentSnapshotInfoCR erstellt, um interne Details zu erhalten.

Beispiel

```
kind: TridentSnapshotInfo
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  snapshotName csi-nas-snapshot
```

- a. Im sekundären Kubernetes-Cluster das Feld *spec.state* der *TridentMirrorRelationship* CR auf *promoted* und *spec.volumeMappings[].promotedSnapshotHandle* auf den *internalName* des Snapshots aktualisieren.
3. Bestätigen Sie auf dem sekundären Kubernetes-Cluster den Status (*status.state*-Feld) von *TridentMirrorRelationship* auf *promoted*.

Eine Spiegelbeziehung nach einem Failover wiederherstellen

Bevor Sie die Spiegelbeziehung wiederherstellen, wählen Sie die Seite aus, die Sie als neue primäre Seite festlegen möchten.

Schritte

1. Stellen Sie auf dem sekundären Kubernetes-Cluster sicher, dass die Werte für das Feld *spec.remoteVolumeHandle* auf der *TridentMirrorRelationship* aktualisiert werden.
2. Auf dem sekundären Kubernetes-Cluster muss das Feld *spec.state* von *TridentMirrorRelationship* auf *reestablished* aktualisiert werden.

Zusätzliche Operationen

Trident unterstützt die folgenden Operationen auf den primären und sekundären Volumes:

Primäres PVC auf ein neues sekundäres PVC replizieren

Stellen Sie sicher, dass Sie bereits ein primäres PVC und ein sekundäres PVC haben.

Schritte

1. Löschen Sie die *PersistentVolumeClaim* und *TridentMirrorRelationship* CRDs aus dem eingerichteten sekundären (Ziel-)Cluster.
2. Löschen Sie die *TridentMirrorRelationship* CRD aus dem primären (Quell-)Cluster.
3. Erstellen Sie eine neue *TridentMirrorRelationship* CRD auf dem primären (Quell-)Cluster für das neue sekundäre (Ziel-)PVC, das Sie einrichten möchten.

Ändern Sie die Größe eines gespiegelten, primären oder sekundären PVC

Die PVC-Größe kann wie gewohnt angepasst werden, ONTAP erweitert automatisch alle Ziel-Flexvols, wenn die Datenmenge die aktuelle Größe überschreitet.

Replikation von einem PVC entfernen

Um die Replikation zu entfernen, führen Sie eine der folgenden Operationen auf dem aktuellen sekundären Volume durch:

- Löschen Sie die *MirrorRelationship* auf dem sekundären PVC. Dadurch wird die Replikationsbeziehung unterbrochen.
- Oder aktualisieren Sie das Feld *spec.state* auf *promoted*.

Löschen Sie ein PVC (das zuvor gespiegelt wurde)

Trident prüft, ob PVCs repliziert sind, und gibt die Replikationsbeziehung frei, bevor versucht wird, das Volume zu löschen.

Löschen eines TMR

Das Löschen eines TMR auf einer Seite einer Spiegelungsbeziehung führt dazu, dass der verbleibende TMR in den Status *promoted* wechselt, bevor Trident den Löschvorgang abschließt. Befindet sich der zum Löschen ausgewählte TMR bereits im Status *promoted*, besteht keine Spiegelungsbeziehung mehr, der TMR wird entfernt und Trident stuft den lokalen PVC auf *ReadWrite* hoch. Durch diese Löschung werden SnapMirror-Metadaten für das lokale Volume in ONTAP freigegeben. Wird dieses Volume zukünftig in einer Spiegelungsbeziehung verwendet, muss beim Erstellen der neuen Spiegelungsbeziehung ein neuer TMR mit einem *established* Volume-Replikationsstatus verwendet werden.

Spiegelbeziehungen aktualisieren, wenn ONTAP online ist

Spiegelungsbeziehungen können jederzeit nach ihrer Einrichtung aktualisiert werden. Sie können die Felder `state: promoted` oder `state: reestablished` verwenden, um die Beziehungen zu aktualisieren. Wenn Sie ein Zielvolume zu einem regulären ReadWrite-Volume hochstufen, können Sie mit *promotedSnapshotHandle* einen bestimmten Snapshot angeben, auf den das aktuelle Volume wiederhergestellt werden soll.

Spiegelbeziehungen aktualisieren, wenn ONTAP offline ist

Sie können eine CRD verwenden, um ein SnapMirror-Update durchzuführen, ohne dass Trident eine direkte Verbindung zum ONTAP-Cluster benötigt. Beachten Sie das folgende Beispielformat von `TridentActionMirrorUpdate`:

Beispiel

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` spiegelt den Status der `TridentActionMirrorUpdate` CRD wider. Es kann den Wert *Succeeded*, *In Progress* oder *Failed* annehmen.

CSI-Topologie verwenden

Trident kann selektiv Volumes erstellen und an Knoten in einem Kubernetes-Cluster anhängen, indem es die ["CSI-Topologiefunktion"](#) verwendet.

Überblick

Mithilfe der CSI-Topologie-Funktion lässt sich der Zugriff auf Volumes auf eine Teilmenge von Knoten beschränken, basierend auf Regionen und Verfügbarkeitszonen. Cloud-Anbieter ermöglichen es Kubernetes-Administratoren heute, zonenbasierte Knoten zu erstellen. Knoten können sich in verschiedenen Verfügbarkeitszonen innerhalb einer Region oder über verschiedene Regionen hinweg befinden. Um die Bereitstellung von Volumes für Workloads in einer Multi-Zonen-Architektur zu erleichtern, verwendet Trident die CSI-Topologie.



Erfahren Sie mehr über die CSI Topology-Funktion ["hier"](#).

Kubernetes bietet zwei einzigartige Volume-Bindungsmodi:

- Mit `VolumeBindingMode` auf `Immediate` gesetzt, erstellt Trident das Volume ohne jegliche Topologiebewusstheit. Volume-Bindung und dynamische Bereitstellung erfolgen, wenn das PVC erstellt wird. Dies ist die Standardeinstellung `VolumeBindingMode` und eignet sich für Cluster, die keine Topologiebeschränkungen erzwingen. Persistente Volumes werden erstellt, ohne von den Planungsanforderungen des anfordernden Pods abhängig zu sein.
- Mit `VolumeBindingMode` auf `WaitForFirstConsumer` gesetzt, wird die Erstellung und Bindung eines Persistent Volume für eine PVC verzögert, bis ein Pod, der die PVC verwendet, geplant und erstellt wird. So werden Volumes erstellt, um die durch Topologieanforderungen erzwungenen Planungsbeschränkungen zu erfüllen.



Der `WaitForFirstConsumer` Bindungsmodus benötigt keine Topologiebezeichnungen. Dies kann unabhängig von der CSI-Topologiefunktion verwendet werden.

Was Sie benötigen

Um die CSI Topology nutzen zu können, benötigen Sie Folgendes:

- Ein Kubernetes-Cluster, der einen ["Unterstützte Kubernetes-Version"](#)

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedaafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedaafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Die Knoten im Cluster sollten Labels aufweisen, die Topologiebewusstsein ermöglichen (`topology.kubernetes.io/region` und `topology.kubernetes.io/zone`). Diese Labels **müssen auf den Knoten im Cluster vorhanden sein**, bevor Trident installiert wird, damit Trident topologiebewusst funktioniert.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{ "\n"}{end}' | grep --color "topology.kubernetes.io"
[nodel1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"nodel1","kubernetes.io/os":"linux","node-role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-a"}]
```

```
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kubernetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/os":"linux","node-role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Schritt 1: Erstellen eines topologiebewussten Backends

Trident-Speicher-Backends können so konzipiert werden, dass sie Volumes selektiv basierend auf Verfügbarkeitszonen bereitstellen. Jedes Backend kann einen optionalen `supportedTopologies` Block enthalten, der eine Liste der unterstützten Zonen und Regionen repräsentiert. Für `StorageClasses`, die ein solches Backend verwenden, würde ein Volume nur erstellt werden, wenn es von einer Anwendung angefordert wird, die in einer unterstützten Region/Zone geplant ist.

Hier ist ein Beispiel für eine Backend-Definition:

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-a
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
```

```

"svm": "iscsi_svm",
"username": "admin",
"password": "password",
"supportedTopologies": [
  {
    "topology.kubernetes.io/region": "us-east1",
    "topology.kubernetes.io/zone": "us-east1-a"
  },
  {
    "topology.kubernetes.io/region": "us-east1",
    "topology.kubernetes.io/zone": "us-east1-b"
  }
]
}

```



supportedTopologies wird verwendet, um eine Liste von Regionen und Zonen pro Backend bereitzustellen. Diese Regionen und Zonen stellen die Liste der zulässigen Werte dar, die in einer StorageClass angegeben werden können. Für StorageClasses, die eine Teilmenge der in einem Backend bereitgestellten Regionen und Zonen enthalten, erstellt Trident ein Volume auf dem Backend.

Sie können supportedTopologies pro Speicherpool ebenfalls definieren. Siehe das folgende Beispiel:

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-centrall
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-centrall
    topology.kubernetes.io/zone: us-centrall-a
  - topology.kubernetes.io/region: us-centrall
    topology.kubernetes.io/zone: us-centrall-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-centrall
        topology.kubernetes.io/zone: us-centrall-a
  - labels:
      workload: dev
    supportedTopologies:

```

- `topology.kubernetes.io/region: us-central1`
`topology.kubernetes.io/zone: us-central1-b`

In diesem Beispiel stehen die `region` und `zone` Bezeichnungen für den Speicherort des Speicherpools. `topology.kubernetes.io/region` und `topology.kubernetes.io/zone` legen fest, von wo aus auf die Speicherpools zugegriffen werden kann.

Schritt 2: Definieren Sie StorageClasses, die topologiebewusst sind

Anhand der den Knoten im Cluster zugewiesenen Topologiebezeichnungen können StorageClasses definiert werden, die Topologieinformationen enthalten. Dadurch werden die Speicherpools bestimmt, die als Kandidaten für PVC-Anfragen dienen, sowie die Teilmenge der Knoten, die die von Trident bereitgestellten Volumes nutzen können.

Siehe das folgende Beispiel:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

In der oben angegebenen StorageClass-Definition wird `volumeBindingMode` auf `WaitForFirstConsumer` gesetzt. PVCs, die mit dieser StorageClass angefordert werden, werden erst verarbeitet, wenn sie in einem Pod referenziert werden. Und `allowedTopologies` gibt die zu verwendenden Zonen und die Region an. Die `netapp-san-us-east1` StorageClass erstellt PVCs auf dem `san-backend-us-east1` oben definierten Backend.

Schritt 3: Eine PVC erstellen und verwenden

Nachdem die StorageClass erstellt und einem Backend zugeordnet wurde, können Sie jetzt PVCs erstellen.

Siehe das Beispiel `spec` unten:

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1

```

Das Erstellen eines PVC mit diesem Manifest würde zu Folgendem führen:

```

kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc

```

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS
pvc-san	Pending				netapp-san-us-east1

```

2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type      Reason              Age   From
  ----      -
  Normal    WaitForFirstConsumer  6s    persistentvolume-controller
  waiting for first consumer to be created before binding

```

Um mit Trident ein Volume zu erstellen und es an das PVC zu binden, verwenden Sie das PVC in einem Pod. Siehe das folgende Beispiel:

```

apiVersion: v1

```

```

kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/region
                operator: In
                values:
                  - us-east1
            preferredDuringSchedulingIgnoredDuringExecution:
              - weight: 1
                preference:
                  matchExpressions:
                    - key: topology.kubernetes.io/zone
                      operator: In
                      values:
                        - us-east1-a
                        - us-east1-b
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
        fsGroup: 2000
    volumes:
      - name: voll
        persistentVolumeClaim:
          claimName: pvc-san
    containers:
      - name: sec-ctx-demo
        image: busybox
        command: [ "sh", "-c", "sleep 1h" ]
        volumeMounts:
          - name: voll
            mountPath: /data/demo
        securityContext:
          allowPrivilegeEscalation: false

```

Dieses podSpec weist Kubernetes an, den Pod auf Knoten zu planen, die in der `us-east1` Region vorhanden sind, und aus beliebigen Knoten auszuwählen, die in der `us-east1-a` oder `us-east1-b` Zone vorhanden sind.

Siehe die folgende Ausgabe:

```
kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE READINESS GATES
app-pod-1     1/1     Running   0           19s   192.168.25.131  node2
<none>        <none>
kubectl get pvc -o wide
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS          AGE   VOLUMEMODE
pvc-san       Bound     pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b  300Mi
RWO           netapp-san-us-east1   48s   Filesystem
```

Backends aktualisieren, um `supportedTopologies` einzuschließen

Vorhandene Backends können aktualisiert werden, um eine Liste von `supportedTopologies` mit `tridentctl backend update` einzuschließen. Dies hat keine Auswirkungen auf bereits bereitgestellte Volumes und wird nur für nachfolgende PVCs verwendet.

Weitere Informationen

- ["Ressourcen für Container verwalten"](#)
- ["nodeSelector"](#)
- ["Affinität und Anti-Affinität"](#)
- ["Taints und Tolerations"](#)

Arbeiten mit Trident Snapshots

Kubernetes-Volume-Snapshots von Persistent Volumes (PVs) ermöglichen zeitpunktgenaue Kopien von Volumes. Sie können einen Snapshot eines mit Trident erstellten Volumes erstellen, einen außerhalb von Trident erstellten Snapshot importieren, ein neues Volume aus einem vorhandenen Snapshot erstellen und Volume-Daten aus Snapshots wiederherstellen.

Überblick

Volume-Snapshot wird von `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` und `google-cloud-netapp-volumes` Treibern unterstützt.

Bevor Sie beginnen

Sie benötigen einen externen Snapshot-Controller und benutzerdefinierte Ressourcendefinitionen (CRDs), um mit Snapshots zu arbeiten. Dies ist die Aufgabe des Kubernetes-Orchestrators (zum Beispiel: Kubeadm, GKE, OpenShift).

Falls Ihre Kubernetes-Distribution den Snapshot-Controller und die CRDs nicht enthält, lesen Sie [Einen Volume Snapshot Controller bereitstellen](#).



Erstellen Sie keinen Snapshot-Controller, wenn Sie in einer GKE-Umgebung bedarfsgesteuerte

Volume-Snapshots erstellen. GKE verwendet einen integrierten, versteckten Snapshot-Controller.

Erstellen Sie einen Volume-Snapshot

Schritte

1. Erstellen Sie ein `VolumeSnapshotClass`. Weitere Informationen finden Sie unter ["VolumeSnapshotClass"](#).
 - Der `driver` verweist auf den Trident CSI driver.
 - `deletionPolicy` kann `Delete` oder `Retain` sein. Wenn auf `Retain` gesetzt, bleibt der zugrunde liegende physische Snapshot auf dem Speichercluster auch dann erhalten, wenn das `VolumeSnapshot` Objekt gelöscht wird.

Beispiel

```
cat snap-sc.yaml
```

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

2. Erstellen Sie einen Snapshot eines bestehenden PVC.

Beispiele

- Dieses Beispiel erstellt eine Snapshot eines bestehenden PVC.

```
cat snap.yaml
```

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

- Dieses Beispiel erstellt ein `Volume-Snapshot`-Objekt für eine PVC mit dem Namen `pvc1` und der Name des Snapshots wird auf `pvc1-snap` gesetzt. Ein `VolumeSnapshot` ist analog zu einer PVC und ist mit einem `VolumeSnapshotContent` Objekt verknüpft, das den eigentlichen Snapshot darstellt.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                                AGE
pvc1-snap                          50s
```

- Sie können das `VolumeSnapshotContent` Objekt für das `pvc1-snap` `VolumeSnapshot` identifizieren, indem Sie es beschreiben. Das `Snapshot Content Name` identifiziert das `VolumeSnapshotContent` Objekt, das diesem Snapshot dient. Der `Ready To Use` Parameter zeigt an, dass der Snapshot zur Erstellung eines neuen PVC verwendet werden kann.

```
kubectl describe volumesnapshots pvc1-snap
Name:          pvc1-snap
Namespace:     default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

Erstellen Sie eine PVC aus einem Volume-Snapshot

Sie können `dataSource` verwenden, um eine PVC mithilfe eines `VolumeSnapshot` mit dem Namen `<pvc-name>` als Datenquelle zu erstellen. Nachdem die PVC erstellt wurde, kann sie einem Pod zugeordnet und wie jede andere PVC verwendet werden.



Die PVC wird im selben Backend wie das Quellvolume erstellt. Weitere Informationen enthält ["KB: Die Erstellung eines PVC aus einem Trident PVC Snapshot kann in einem alternativen Backend nicht durchgeführt werden"](#).

Im folgenden Beispiel wird die PVC unter Verwendung von `pvc1-snap` als Datenquelle erstellt.

```
cat pvc-from-snap.yaml
```

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

Importieren eines Volume-Snapshots

Trident unterstützt die "[Kubernetes pre-provisioned Snapshot-Prozess](#)", damit der Cluster-Administrator ein VolumeSnapshotContent-Objekt erstellen und außerhalb von Trident erstellte Snapshots importieren kann.

Bevor Sie beginnen

Trident muss das übergeordnete Volume des Snapshots erstellt oder importiert haben.

Schritte

1. **Cluster-Administrator:** Erstellen Sie ein VolumeSnapshotContent Objekt, das auf den Backend-Snapshot verweist. Dadurch wird der Snapshot-Workflow in Trident gestartet.
 - Geben Sie den Namen des Backend-Snapshots in annotations als `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">` an.
 - Geben Sie `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` in `snapshotHandle` an. Dies ist die einzige Information, die Trident vom externen Snapshotter im `ListSnapshots`-Aufruf bereitgestellt wird.



Der `<volumeSnapshotContentName>` kann aufgrund von Namensbeschränkungen der CRs nicht immer mit dem Backend-Snapshot-Namen übereinstimmen.

Beispiel

Das folgende Beispiel erstellt ein VolumeSnapshotContent Objekt, das auf einen Backend-Snapshot `snap-01` verweist.

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:

```

```

    trident.netapp.io/internalSnapshotName: "snap-01" # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default

```

2. **Cluster-Administrator:** Erstellen Sie das VolumeSnapshot CR, das auf das VolumeSnapshotContent Objekt verweist. Dadurch wird der Zugriff auf die Verwendung des VolumeSnapshot in einem bestimmten Namespace angefordert.

Beispiel

Das folgende Beispiel erstellt ein VolumeSnapshot CR mit dem Namen import-snap, das auf das VolumeSnapshotContent mit dem Namen import-snap-content verweist.

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content

```

3. **Interne Verarbeitung (keine Aktion erforderlich):** Der externe Snapshotter erkennt den neu erstellten VolumeSnapshotContent und führt den ListSnapshots-Aufruf aus. Trident erstellt das TridentSnapshot.
 - Der externe Snapshotter setzt das VolumeSnapshotContent auf readyToUse und das VolumeSnapshot auf true.
 - Trident kehrt zurück readyToUse=true.
4. **Beliebiger Benutzer:** Erstellen Sie ein PersistentVolumeClaim, um auf das neue VolumeSnapshot zu verweisen, wobei der spec.dataSource (oder spec.dataSourceRef) Name der VolumeSnapshot Name ist.

Beispiel

Das folgende Beispiel erstellt eine PVC, die auf die VolumeSnapshot mit dem Namen import-snap verweist.

```

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

Stellen Sie Volumendaten mithilfe von Snapshots wieder her

Das Snapshot-Verzeichnis ist standardmäßig ausgeblendet, um maximale Kompatibilität von Volumes zu gewährleisten, die mit den `ontap-nas` und `ontap-nas-economy`-Treibern bereitgestellt wurden. Aktivieren Sie das `.snapshot`-Verzeichnis, um Daten direkt aus Snapshots wiederherzustellen.

Verwenden Sie die ONTAP CLI `volume snapshot restore`, um ein Volume auf einen Zustand wiederherzustellen, der in einem früheren Snapshot aufgezeichnet wurde.

```

cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive

```



Beim Wiederherstellen einer Snapshot-Kopie wird die bestehende Volume-Konfiguration überschrieben. Änderungen an den Volume-Daten, die nach der Erstellung der Snapshot-Kopie vorgenommen wurden, gehen verloren.

In-Place-Volume-Wiederherstellung aus einem Snapshot

Trident ermöglicht die schnelle, direkte Wiederherstellung von Volumes aus einem Snapshot mithilfe des `TridentActionSnapshotRestore` (TASR) CR. Dieser CR fungiert als imperative Kubernetes-Aktion und bleibt nach Abschluss des Vorgangs nicht erhalten.

Trident unterstützt die Snapshot-Wiederherstellung auf den `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` und `solidfire-san`-Treibern.

Bevor Sie beginnen

Sie müssen ein gebundenes PVC und einen verfügbaren Volume Snapshot haben.

- Überprüfen Sie, ob der PVC-Status gebunden ist.

```
kubectl get pvc
```

- Prüfen Sie, ob der Volume-Snapshot einsatzbereit ist.

```
kubectl get vs
```

Schritte

1. Erstellen Sie den TASR CR. Dieses Beispiel erstellt einen CR für PVC `pvc1` und Volume Snapshot `pvc1-snapshot`.



Der TASR CR muss sich in einem Namespace befinden, in dem der PVC und der VS existieren.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Wenden Sie die CR an, um aus dem Snapshot wiederherzustellen. Dieses Beispiel stellt aus dem Snapshot `pvc1` wieder her.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Ergebnisse

Trident stellt die Daten aus dem Snapshot wieder her. Sie können den Status der Snapshot-Wiederherstellung überprüfen:

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
```

```

items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""

```



- In den meisten Fällen wird Trident den Vorgang bei einem Fehler nicht automatisch wiederholen. Sie müssen den Vorgang erneut ausführen.
- Kubernetes-Benutzer ohne Administratorrechte müssen möglicherweise vom Administrator die Berechtigung erhalten, einen TASR CR in ihrem Anwendungs-Namespace zu erstellen.

Löschen eines PV mit zugehörigen Snapshots

Beim Löschen eines Persistent Volume mit zugehörigen Snapshots wird das entsprechende Trident volume in den Status „Deleting state“ versetzt. Entfernen Sie die Volume-Snapshots, um das Trident volume zu löschen.

Einen Volume Snapshot Controller bereitstellen

Falls Ihre Kubernetes-Distribution den Snapshot-Controller und die CRDs nicht enthält, können Sie sie wie folgt bereitstellen.

Schritte

1. Erstellen Sie Volume Snapshot-CRDs.

```
cat snapshot-setup.sh
```

```

#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-

```

```
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Erstellen Sie den Snapshot Controller.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```



Öffnen Sie bei Bedarf `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` und aktualisieren Sie namespace auf Ihren Namespace.

Verwandte Links

- ["Volume Snapshots"](#)
- ["VolumeSnapshotClass"](#)

Arbeiten mit Trident Volume-Gruppen-Snapshots

Kubernetes-Volume-Gruppen-Snapshots von Persistent Volumes (PVs) NetApp Trident bietet die Möglichkeit, Snapshots mehrerer Volumes (einer Gruppe von Volume-Snapshots) zu erstellen. Dieser Volume-Gruppen-Snapshot stellt Kopien mehrerer Volumes dar, die zum gleichen Zeitpunkt erstellt wurden.



VolumeGroupSnapshot ist eine Beta-Funktion in Kubernetes mit Beta-APIs. Kubernetes 1.32 ist die Mindestversion, die für VolumeGroupSnapshot erforderlich ist.

Snapshots von Volumengruppen erstellen

Snapshot für Volumengruppen wird mit den folgenden Speichertreibern unterstützt:

- `ontap-san` Treiber – nur für die iSCSI- und FC-Protokolle, nicht für das NVMe/TCP-Protokoll.
- `ontap-san-economy` - nur für das iSCSI-Protokoll.

- `ontap-nas`



Volume-Gruppen-Snapshot wird für NetApp ASA r2 oder AFX-Speichersysteme nicht unterstützt.

Bevor Sie beginnen

- Stellen Sie sicher, dass Ihre Kubernetes-Version K8s 1.32 oder höher ist.
- Sie benötigen einen externen Snapshot-Controller und benutzerdefinierte Ressourcendefinitionen (CRDs), um mit Snapshots zu arbeiten. Dies ist die Aufgabe des Kubernetes-Orchestrators (zum Beispiel: Kubeadm, GKE, OpenShift).

Falls Ihre Kubernetes-Distribution den externen Snapshot-Controller und die CRDs nicht enthält, lesen Sie [Einen Volume Snapshot Controller bereitstellen](#).



Erstellen Sie keinen Snapshot-Controller, wenn Sie in einer GKE-Umgebung bedarfsgesteuerte Snapshots von Volumengruppen erstellen. GKE verwendet einen integrierten, versteckten Snapshot-Controller.

- Setzen Sie im Snapshot-Controller-YAML das `CSIVolumeGroupSnapshot` Feature-Gate auf 'true', um sicherzustellen, dass Volume Group Snapshot aktiviert ist.
- Erstellen Sie die erforderlichen Volume Group Snapshot-Klassen, bevor Sie einen Volume Group Snapshot erstellen.
- Stellen Sie sicher, dass sich alle PVCs/Volumes auf demselben SVM befinden, um VolumeGroupSnapshot erstellen zu können.

Schritte

- Erstellen Sie eine `VolumeGroupSnapshotClass`, bevor Sie eine `VolumeGroupSnapshot` erstellen. Weitere Informationen finden Sie unter ["VolumeGroupSnapshotClass"](#).

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Erstellen Sie PVCs mit den erforderlichen Labels unter Verwendung vorhandener Storage Classes, oder fügen Sie diese Labels zu bestehenden PVCs hinzu.

Das folgende Beispiel erstellt die PVC unter Verwendung von `pvc1-group-snap` als Datenquelle und der Bezeichnung `consistentGroupSnapshot: groupA`. Definieren Sie den Bezeichnungsschlüssel und -wert entsprechend Ihren Anforderungen.

```
kind: PersistentVolumeClaim
apiVersion: v1
```

```

metadata:
  name: pvcl-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
    storageClassName: sc1-1

```

- Erstellen Sie ein VolumeGroupSnapshot mit der gleichen Bezeichnung (consistentGroupSnapshot: groupA wie im PVC angegeben).

Dieses Beispiel erstellt einen Snapshot einer Volumengruppe:

```

apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA

```

Volumendaten mithilfe eines Gruppen-Snapshots wiederherstellen

Sie können einzelne Persistent Volumes mithilfe der einzelnen Snapshots wiederherstellen, die als Teil des Volume Group Snapshot erstellt wurden. Sie können den Volume Group Snapshot nicht als Einheit wiederherstellen.

Verwenden Sie die ONTAP CLI `volume snapshot restore`, um ein Volume auf einen Zustand wiederherzustellen, der in einem früheren Snapshot aufgezeichnet wurde.

```

cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive

```



Beim Wiederherstellen einer Snapshot-Kopie wird die bestehende Volume-Konfiguration überschrieben. Änderungen an den Volume-Daten, die nach der Erstellung der Snapshot-Kopie vorgenommen wurden, gehen verloren.

In-Place-Volume-Wiederherstellung aus einem Snapshot

Trident ermöglicht die schnelle, direkte Wiederherstellung von Volumes aus einem Snapshot mithilfe des `TridentActionSnapshotRestore` (TASR) CR. Dieser CR fungiert als imperative Kubernetes-Aktion und bleibt nach Abschluss des Vorgangs nicht erhalten.

Weitere Informationen finden Sie unter ["In-Place-Volume-Wiederherstellung aus einem Snapshot"](#).

Löschen eines PV mit zugehörigen Gruppensnapshots

Beim Löschen eines Gruppenvolume-Snapshots:

- Sie können VolumeGroupSnapshots als Ganzes löschen, nicht einzelne Snapshots in der Gruppe.
- Wenn PersistentVolumes gelöscht werden, während ein Snapshot für dieses PersistentVolume existiert, versetzt Trident dieses Volume in den Status „wird gelöscht“, da der Snapshot entfernt werden muss, bevor das Volume sicher entfernt werden kann.
- Wenn ein Klon mithilfe eines gruppierten Snapshots erstellt wurde und die Gruppe anschließend gelöscht werden soll, wird ein Split-on-Clone-Vorgang gestartet und die Gruppe kann erst gelöscht werden, wenn der Split abgeschlossen ist.

Einen Volume Snapshot Controller bereitstellen

Falls Ihre Kubernetes-Distribution den Snapshot-Controller und die CRDs nicht enthält, können Sie sie wie folgt bereitstellen.

Schritte

1. Erstellen Sie Volume Snapshot-CRDs.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcontents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.yaml
```

2. Erstellen Sie den Snapshot Controller.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-controller/setup-snapshot-controller.yaml
```



Öffnen Sie bei Bedarf `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` und aktualisieren Sie namespace auf Ihren Namespace.

Verwandte Links

- ["VolumeGroupSnapshotClass"](#)
- ["Volume Snapshots"](#)

Copyright-Informationen

Copyright © 2026 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtsinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

Software, die von urheberrechtlich geschütztem NetApp Material abgeleitet wird, unterliegt der folgenden Lizenz und dem folgenden Haftungsausschluss:

DIE VORLIEGENDE SOFTWARE WIRD IN DER VORLIEGENDEN FORM VON NETAPP ZUR VERFÜGUNG GESTELLT, D. H. OHNE JEGLICHE EXPLIZITE ODER IMPLIZITE GEWÄHRLEISTUNG, EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE STILLSCHWEIGENDE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, DIE HIERMIT AUSGESCHLOSSEN WERDEN. NETAPP ÜBERNIMMT KEINERLEI HAFTUNG FÜR DIREKTE, INDIREKTE, ZUFÄLLIGE, BESONDERE, BEISPIELHAFTE SCHÄDEN ODER FOLGESCHÄDEN (EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE BESCHAFFUNG VON ERSATZWAREN ODER -DIENSTLEISTUNGEN, NUTZUNGS-, DATEN- ODER GEWINNVERLUSTE ODER UNTERBRECHUNG DES GESCHÄFTSBETRIEBS), UNABHÄNGIG DAVON, WIE SIE VERURSACHT WURDEN UND AUF WELCHER HAFTUNGSTHEORIE SIE BERUHEN, OB AUS VERTRAGLICH FESTGELEGTER HAFTUNG, VERSCHULDENSUNABHÄNGIGER HAFTUNG ODER DELIKTSHAFTUNG (EINSCHLIESSLICH FAHRLÄSSIGKEIT ODER AUF ANDEREM WEGE), DIE IN IRGEND EINER WEISE AUS DER NUTZUNG DIESER SOFTWARE RESULTIEREN, SELBST WENN AUF DIE MÖGLICHKEIT DERARTIGER SCHÄDEN HINGEWIESEN WURDE.

NetApp behält sich das Recht vor, die hierin beschriebenen Produkte jederzeit und ohne Vorankündigung zu ändern. NetApp übernimmt keine Verantwortung oder Haftung, die sich aus der Verwendung der hier beschriebenen Produkte ergibt, es sei denn, NetApp hat dem ausdrücklich in schriftlicher Form zugestimmt. Die Verwendung oder der Erwerb dieses Produkts stellt keine Lizenzierung im Rahmen eines Patentrechts, Markenrechts oder eines anderen Rechts an geistigem Eigentum von NetApp dar.

Das in diesem Dokument beschriebene Produkt kann durch ein oder mehrere US-amerikanische Patente, ausländische Patente oder anhängige Patentanmeldungen geschützt sein.

ERLÄUTERUNG ZU „RESTRICTED RIGHTS“: Nutzung, Vervielfältigung oder Offenlegung durch die US-Regierung unterliegt den Einschränkungen gemäß Unterabschnitt (b)(3) der Klausel „Rights in Technical Data – Noncommercial Items“ in DFARS 252.227-7013 (Februar 2014) und FAR 52.227-19 (Dezember 2007).

Die hierin enthaltenen Daten beziehen sich auf ein kommerzielles Produkt und/oder einen kommerziellen Service (wie in FAR 2.101 definiert) und sind Eigentum von NetApp, Inc. Alle technischen Daten und die Computersoftware von NetApp, die unter diesem Vertrag bereitgestellt werden, sind gewerblicher Natur und wurden ausschließlich unter Verwendung privater Mittel entwickelt. Die US-Regierung besitzt eine nicht ausschließliche, nicht übertragbare, nicht unterlizenzierbare, weltweite, limitierte unwiderrufliche Lizenz zur Nutzung der Daten nur in Verbindung mit und zur Unterstützung des Vertrags der US-Regierung, unter dem die Daten bereitgestellt wurden. Sofern in den vorliegenden Bedingungen nicht anders angegeben, dürfen die Daten ohne vorherige schriftliche Genehmigung von NetApp, Inc. nicht verwendet, offengelegt, vervielfältigt, geändert, aufgeführt oder angezeigt werden. Die Lizenzrechte der US-Regierung für das US-Verteidigungsministerium sind auf die in DFARS-Klausel 252.227-7015(b) (Februar 2014) genannten Rechte beschränkt.

Markeninformationen

NETAPP, das NETAPP Logo und die unter <http://www.netapp.com/TM> aufgeführten Marken sind Marken von NetApp, Inc. Andere Firmen und Produktnamen können Marken der jeweiligen Eigentümer sein.